Institut für Architektur von Anwendungssystemen

Universität Stuttgart

Universitätsstraße 38

D–70569 Stuttgart

Master Thesis No. 3348
Cloud Migration – a Reference Process Model

Anand Babu Ramalingam

Study Course:	INFOTECH
Examiner:	Prof. Dr. Frank Leymann
Supervisors:	Alexander Nowak, Sebastian Wagner
Started at:	2012-06-01
Finished at:	2012-11-30
CR-Classification:	H.4.1, H.3.5, D.2.1, D.2.2, D.2.13,

Abstract

Cloud computing is being rapidly adopted by both public and private sector organizations. However, Migration of existing applications into a cloud environment is not trivial; and it involves multiple facets.While the functional requirements are handled by the IT experts, the non-functional and organizational aspects like security, privacy, complaince, etc. need to be addressed by the appropriate stakeholders. This is a comprehensive and involved process. The stakeholders need to be exhaustive in their research and planning to ensure a smooth and successful migration.

In this work, we propose a generic, model-driven, platform independent process model that will help the different stakeholders in the enterprise to take a systematic approach in their journey towards cloud migration. It does so by ensuring both IT and business experts work colloboratively in an efficient manner to avoid potential pitfalls. The issues with different types of cloud migration scenarios are addressed with this generic process model.

The proposed process model is validated by applying it to three different real-life cloud migration scenarios.

Keywords: BPMN2.0, Cloud computing, Business Process Re-engineering, Model Driven Engineering, Service Oriented Architecture.

List of Tables

List of Figures

Table of Contents

1. Introduction

1.1. Motivation

Information and communication technology makes life on our planet easier in many different ways and opens up several new possibilities both for our work and private lives. There is an unprecedented trend in digitization of business processes in many industries like banking, e-government, e-health, digital entertainment, etc. [70]. With the evolution and application of these computing technologies, number of people consuming multiple services continues to increase exponentially. There is already a perception that computing will one day be the 5th utility (after water, electricity, gas and telephony) which will provide the basic level of computing services that are essential to meet the everyday needs of a general community.With such a demand for consumption, despite the technological developments and the more and more energy-efficient devices, the energy requirement for Information and Communication Technologies will continue to increase [69]. In fact, the worldwide data center CO2 emissions is about half of the overall airlines emissions as well as being greater than both the Argentina and the Netherlands emissions [70].

Many companies, public as well as private realize the operational inefficiencies of their datacenters which impacts both their operating expenses in addition to huge capital expenditures. With Computing available as a service in a cloud computing paradigm, many companies consider Cloud Computing as a solution to procure services. Therefore, more and more applications will be migrated to the cloud.

Further in most of the traditional IT environment, information systems of each department are organized in a "silo" model, which further leads to operational inefficiency. Though a shared services model have been proposed and pursued, it has not been implemented in an efficient way across different IT organizations. Cloud computing technologies can enable these organizations to implement the shared services model efficiently. Enterprises, both public and private have already made early moves into cloud computing and the technology of cloud computing has been proved by different projects already. For e.g. in Washington D.C all city government employees have adopted Google documents and services such as gmail [67]. The U.S. General Services Administration has announced moving the entire government wide portal usa.gov to the cloud [68]. In Japan by 2015, the Ministry of industrial affairs and communications plans to shift all government agencies into a private cloud environment [44]. However, these successes have come from ad-hoc strategies that are particular for that migration scenario. There is a need to develop process models, methods, tools and design patterns for the systematic and targeted migration of ICT systems to more efficient cloud environments. For this, one needs to understand the processes in the current environment,

design appropriate cloud computing solutions and come up with a model that guides in migrating to the cloud environment.

On the other hand, inspite of the numerous opportunities that Cloud computing provides, it introduces new challenges that needs to be understood to complete a successful migration. These issues affect IT as well as the businesses. However, since most of the cloud computing migration projects are driven by the IT, these problems are not effectively addressed. Hence, there is a need for a process model that provides a framework to allow IT and businesses to collaborate effectively to tackle these challenges.

1.2. Problem Statement

Migration of existing applications into cloud environment is not trivial and requires manifold aspects depending on the different application components and non-functional properties. In some cases, it might be suitable to migrate the whole application stack from the local data center into the cloud, while in other scenarios already existing cloud offerings might replace specific application components.

The main objective of this thesis is to develop a comprehensive reference process model that supports stakeholders to migrate existing applications into the cloud. The objectives include the following

O1: Investigate existing solutions and best-practices of Cloud migration scenarios
O2: Develop a reference process model for cloud migration
O3: Prove that the process model works for all types of applications to different cloud environments by providing sample use cases.

1.3. Approach

We began by breaking the problem domain into different sub-sets which can then be handled effectively.

Migration to Cloud computing is motivated by problems that occur at different levels in the enterprise. They can be viewed as platform level problems (physical server level) or data and application level platforms (application architecture level) or business-level problems(external situations, new business strategy).

1.3.1. Platform level problems

Information systems, both in private and public sectors are subject to low resource utilization with environments that are difficult to manage. The platform on which the information system is deployed is also not in line with the business objectives.

The existing physical assets are not utilized completely resulting in energy loss, higher maintenance cost, etc.. The problem is further compounded as IT needs to respond to business requirements in an agile manner, which further leads to an even more in-efficient IT architecture.

1.3.2. Data and architecture level problems

Existing data and application architectures are not run in direct alignment with modern IT disciplines [45]. These environments often consists of plenty of legacy systems which are difficult to understand, costly to maintain and almost impossible to make changes. Most of the legacy systems cannot be changed at the pace that the businesses might require. IT was an enabler to solve business problems with technology. Over time, the technology has become a constraint. IT needs to migrate to an environment that allows it to respond to business changes effectively.

1.3.3. Business level problems

Enterprises are facing increasing pressures from market situations, competitions, etc. There is a need to reduce the capital expenditure (capex) as well as operating expenditure (opex) as well as IT staffing [5].

Organizations are trying to reduce risk that is inherent while buying a contract for an IT system/software and the value it generates.

The businesses need a shorter time-to-market for new products and services. It needs to avail services in a customer centric fashion (e.g. Web, Mobile, Other PDA, etc.).

Both, IT and business challenges require modernization of IT [45]. The challenge is to migrate these physical and software assets efficiently while it supports the changing businesses.

IT has the potential to reduce information and communication technology costs by virtualizing capital assets like disk storage and processing cycles into readily available, affordable operating expense.

Cloud computing's paradigm addresses many of the above concerns. For e.g. the pay-as-you go model of Cloud Computing addresses the capex and opex concerns of organizations. It is enough to pay for only the features that an organization is interested and the amount it uses, instead of investing heavily on building a new datacenter or buying a whole suite of enterprise software and implementing in-house.

Craig et al. [44] identifies that one of the primary motivations towards cloud computing among public enterprises is the ability to share resources among multiple agencies. They

further add that most of the information management in traditional IT Service Centers are organized in the form of "silos" where the organizations operate in a stand-alone manner.

1.3.4. Cloud Computing Problems

Though, Cloud computing solves many of the above problems, we need to be aware of the potential problems that cloud computing brings into the table.

Government's efforts to create shared services have not been successful. Craig et al. [44] reasons that lack of appropriate incentives or sanctions, overprotection of budget and resources as the main reasons behind this behavior. Most often there have been instances where the cost of using shared services can exceed the cost of self-provisioning those services. Also, often, the shared services model does not result in the projected savings.

Further, Craig et al. [44] identifies the following issues with cloud computing.

- The Cloud provider might not be directly under the direct jurisdiction of the government. This might introduces risk in terms of access to those services as well as on security.

- Further, Open standards and interoperability are not always supported by the enterprises. Since the organizations have already invested on these open standards, this might be detrimental in their cloud adoption.

- Data privacy is another concern while using public clouds. Many national/international data protection laws prohibit certain organizations from using these public services. A list of these laws has been listed in [50].

There are governance and management challenges in adopting cloud computing [44]. There must be assurances from the service provider to meet the legal and policy constraints while complying with internal and external audit requirements.

Business continuity is a concern. Cloud outages are a reality [71,72].

Whatever may be the problems that drive cloud migration, a generic process model needs to be modeled which can help enterprises of all sizes with the next steps.

We then understood the state of the art in cloud computing platforms, Cloud Computing design patterns and identified patterns from successful cloud migrations. We then studied the migration strategies that have been used in migrating Information systems. We also identified some cloud computing migration case studies which were not successful in reaching their stated objectives.

Since cloud migration required considering manifold aspects from the views of different stakeholders involved in the migration, frameworks for capturing the different views of the organization were identified. They were evaluated to identify activities that are relevant for a generic process model for cloud migration.

We then studied the existing migration strategies for Information systems and the application of those strategies for migrating Information Systems to the cloud. We then identified the necessary activities needed to build a generic process model for cloud migration. The process model was proven by providing different use cases of cloud migration projects.

1.4. Outline

Section 2 defines basic definition for the concepts needed for this work. Section 3 introduces related work in this field. Section 4 introduces to relevant migration strategies. Section 5 describes the resulting process model for cloud migration while section 6 provides use-case examples for this process model. Finally Section 7 presents us with conclusion and future work.

2. Definitions

Workflow [46]

“Workflow is a technology for realizing of inter-/intra-enterprise (business) processes” and

“Workflow constructs allow to implement business process aspects like logical decision points, sequential as wells as parallel work routs, as well as managing of exceptional situations. This is realized by the means of control flow constructs of a workflow language. The business rules (complex transition conditions) specify in reusable manner the way to process the workflow specific data.”

Workflow Management System (WfMS) [39]

“A system that completely defines manages and executes ‘workflows’ through the execution of software whose order of execution is driven by a computer representation of the workflow logic.”

Workflow Engine

“A software service or ‘engine’ that provides the run time execution environment for a workflow instance.” [18]

“The main purpose of the runtime component of the workflow management system is to proactively drive processes. The runtime component navigates through a process model and interacts with users and applications [...].” [40, p. 97]

Cloud Computing

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

This Cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.

“Essential Characteristics: On-demand self-service, Broad network access, Resource pooling, Rapid elasticity, Measured Service”

“Service Models: Cloud Software as a Service (SaaS), Cloud Platform as a Service (PaaS), Cloud Infrastructure as a Service (IaaS)”

“Deployment Models: Private Cloud, Community Cloud, Public Cloud, Hybrid Cloud“[65].

Business Process Management (BPM)

“Business process management (BPM) is a systematic approach to making an organization's workflow more effective, more efficient and more capable of adapting to an ever-changing environment. The goal of BPM is to reduce human error and miscommunication and focus stakeholders on the requirements of their roles. BPM is a subset of infrastructure management, an administrative area concerned with maintaining and optimizing an organization's equipment and core operations.” [66].

Model Driven Architecture (MDA)

Model-Driven Architecture is defined “as an approach to system-specification and interoperability based on the use of formal models OMG [MDA, MDA2 and DSouza]. In MDA, platform-independent models (PIMs) are initially expressed in a platform-independent modeling language, such as UML. The platform-independent model is subsequently translated to a platform-specific model (PSM) by mapping the PIM to some implementation language or platform (e.g. Java) using formal rules.” [64].

Service Oriented Architecture (SOA)

“In software engineering, a service-oriented architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes. SOA design principles are used during the phases of systems development and integration” [75].

Business Re-engineering

The fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service and speed [24].

3. Related Work

In this chapter, we provide other frameworks that are related to this work. These frameworks discuss the necessary activities and the order of these activities needed by the different stakeholders involved. Below, we describe how our process model is related to these frameworks to them while underlining the areas that our process model covers which were left open by these frameworks.

3.1. The Federal Cloud Computing Strategy

Kundra [89] provides a decision framework for cloud migration along with some case examples to illustrate the framework. The framework is flexible and consists of 3 generic phases namely (i) Select (ii) Provision and (iii) Manage.

- The select phase provides guidelines on how to select IT services of value to move while providing guidance on the risks that IT organizations need to be aware of.

- The Provision phase is about how the IT services need to be provisioned in the cloud with guidance on how to aggregate demand for cloud services across different organizations, how to integrate services to create a shared service model, how to contract effectively and realize value from decommissioning/reusing existing infrastructure.

- In the Manage phase, how the services need to be managed is proposed; (i) by monitoring the services actively and (ii) by periodically re-evaluating the services to ensure the services run as expected.

The guidance on organization issues like aggregating demand for cloud usage and the pointers on non-functional issues like the key security considerations provide the necessary motivation on these aspects for the work.

Though the framework is flexible, it does not provide us with a complete process model that lists all the stakeholders and the roles they need to play. It also does not describe the starting point from which an information system can be analyzed as well as the next steps needed to identify systems that can be migrated to the cloud.

In contrast in our process model, we define an approach on how to approach the information system to identifying sources of value that can be leveraged by migrating these systems to the cloud. We also identify the stakeholders involved in the migration and the process model specifies when these stakeholders need to execute what particular step. The guidance on the

organizational and non-functional management provided by this framework is taken into consideration while designing our process model.

3.2. REMICS (REuse and Migration of legacy applications to Interoperable Cloud Services)

REMICS (REuse and Migration of legacy applications to Interoperable Cloud Services) [100] is a European research project on migrating legacy systems to the cloud. In particular, the project's main objective is to develop methods and tools to migrate legacy applications to the "Service Cloud" paradigm. The "Service Cloud" stands for Service Oriented Architecture and Cloud Computing for the development of "Software as a Service (SaaS)" systems.

They provide a framework which is based on Architecture Driven Modernization (ADM) that is developed by OMG.

The framework involves two phases.

- The first phase is called Recover, where the architecture of the legacy system is obtained using reverse engineering standards like Knowledge Discovery Metamodel (KDM). The architecture model of the system is then analyzed to obtain models of business processes, rules, components, implementation and test specification.

- In the next phase called "Migrate", the models from the "Recover phase" form the basis for the development of their corresponding models on the target cloud environment. In this phase, the new architecture will be built based on SOA/Cloud computing platforms.

The whole migration process is supported by two complementary activities: i) "Model-Driven Interoperability (MDI)" to manage interoperability between services and ii) "Validate, Control and Supervise" to guarantee that the migrated system provides the required functionality with the required quality of services.

This project focuses on migrating legacy applications to a "service cloud" for the development of "Software as a Service" system alone. This project does not deal with migrating legacy systems to other cloud service models. This project does not also deal with migrating non-legacy applications like an application that is built already on Service Oriented architecture. This framework does not provide any guidelines regarding the organization and non-functional aspects of the cloud migration. The Architecture Driven Modernization that is used as a baseline concept for this project is only used to migrate legacy applications to "service cloud" paradigm. However, Architecture Driven modernization can be used to migrate all types of applications to different target environments. Our process model makes use of these features of the Architecture Driven Modernization concepts to develop our generic process model for cloud migration.

3.3. Cloud Computing and SOA Convergence in Your Enterprise

Linthicum [19] proposes the steps needed to approach cloud migration using Service Oriented Architecture (SOA) as a strategy. The macro steps defined in this work are

- The business case is defined for migrating applications to the cloud.
- Once the business case is made, the data catalog for the existing architecture is modeled and immediately the corresponding information model of the cloud architecture is identified.
- The candidate services in the problem domain are identified and the service model is built which lists the possible services that can be migrated.
- Similarly, a process model is built with the candidate processes that can be migrated to cloud.
- Next, a governance model is designed which involves defining, designing and implementing policies for design-time and runtime governance for cloud computing.
- A test plan that supports testing from different levels(service level, process level, governance-level) are designed
- The data, service and processes are analyzed and they are marked for migration to the cloud and the end-state architecture is built.
- The target platforms are listed. After evaluating them, a target platform is chosen and the components are deployed on the target platform.

This method imposes SOA as the architecture style on the end-state of the architecture. It is not possible for all the applications to have service oriented architecture in the cloud platform. Also, the different aspects and the stakeholders involved in the migration and the challenges are not identified. The work focusses on the information system to be migrated alone. The steps listed in the method are not complete. For e.g. it's not possible to build a business case without understanding the current state of the information system. Yet as per this method, it's the first step in this migration process. However, we are motivated by the high-level steps that are described in this work.

4. Strategies for Cloud migration

Cloud migration is different from any of the other IT migrations viz. Database migration, Application server migration, Physical Server migration, etc. In all those migrations, the migrated system and the old system had to deal with IT aspects alone. With cloud migration, the migrated system might enable different possibilities (theoretically infinite scalability, elasticity, better resource utilization, Cost efficiency, etc.). However, it also brings with it some issues/concerns like privacy, security, performance that needs to be taken into consideration. Thus, this migration effort involves dimensions that are not restricted to IT alone but also other non-IT departments. Hence, the strategies that are needed to modernize the legacy Information Systems need to tackle these challenges.

Below, we describe a set of software migration strategies, their advantages and disadvantages before we provide a summary of all these strategies withs respect to cloud migration.

4.1. Chicken Little migration Strategy

Brodie and Stonebraker [51] have proposed the Chicken Little strategy in their DARWIN project to provide support for mission critical legacy system migration projects. It is described as an approach to incrementally migrate legacy systems with a constraint. The constraint is that the target and the legacy system must be inter-operable for every increment. This approach of legacy system migration using Chicken Little strategy is motivated by the following facts:

- Specifications for the legacy systems are rarely found. The only documentation available for legacy Information System is the code itself.
- There are number of undocumented dependencies in the legacy ISs.
  Legacy systems are difficult to cut over. Most of the legacy databases or files are too large and it might take weeks to download them. Even if the rewritten IS were fully operational, there are no techniques to migrate the **live data** from the legacy IS to the new IS within the time-frame that the business can support without its mission critical IS.

4.1.1. Principle involved

Chicken Little Strategy involves migrating legacy IS by splitting into small pieces of increments. Each increment requires a small amount of resource, a short time-frame and produces a small result towards the desired goal. The main measure to be considered while planning for the migration is the time required for the entire migration. The financial and business responsibilities of the client should be taken care during the migration plan.

Chicken Little strategy is concerned with selecting the independent increments to be migrated (i.e. small legacy applications, a portion of legacy application, interfaces (or) databases).It is also necessary to take care of the sequence in which the increments are migrated. The problem of dependencies between the increments should be taken care of.

Considering from an architectural point of view, the Information System is a combination of three components: interfaces, applications and database services. These components can be distinguished as the different levels of IS architecture. [52]

Chicken Little strategy relies on two main concepts, namely

- IS architecture decomposability
- Gateways [52].

Information System Architecture Decomposability

Based on the decomposable nature, the legacy systems are categorized into 4 categories:

- Modular legacy systems
- Semi decomposable legacy systems
- Non-Decomposable legacy systems
- Hybrid legacy systems

Modular Legacy systems

In these legacy systems (see Figure 4-1), the three kinds of components are clearly distinguished. They consist of a collection of independent application Modules (Mi). Each of these modules(Mi) interact with a database service and each might have its own user interface(Ui) and/or System level interface(Slk) through which it interacts with one or more IS. These are best suited for the migration process.

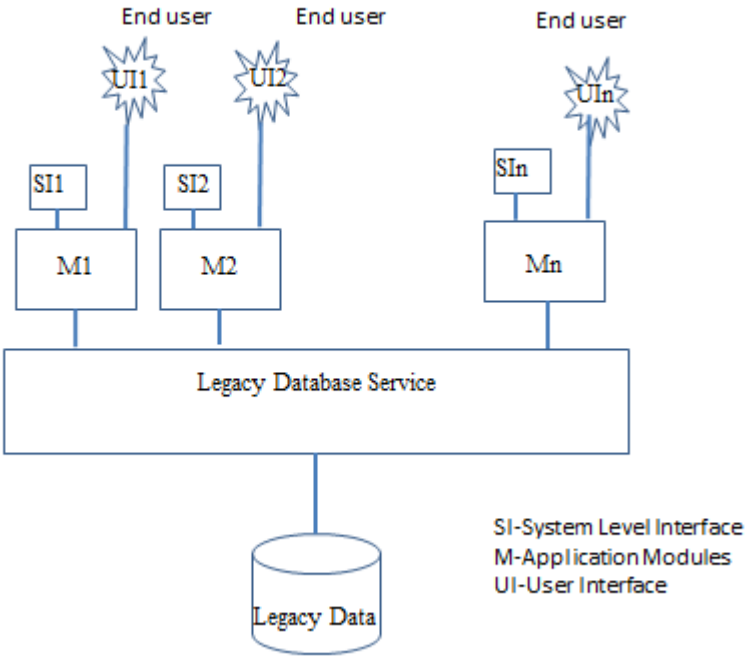


Figure 4-1 Decomposable Legacy IS Architecture [51]

Semi-decomposable legacy systems

At this level, only user and system level interfaces are independent components [52].The lack of understanding about the application modules and the database modules make the migration of legacy systems more complex.

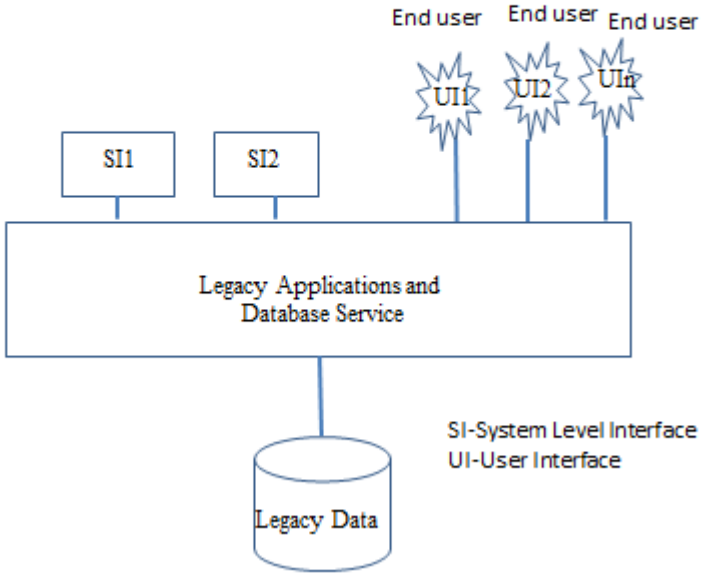


Figure 4-2 Semi-Decomposable Legacy IS Architecture [51]

Non-Decomposable legacy systems

These legacy systems do not reveal the components of the architecture explicitly. It is seen as a black box since no component can be clearly separated.

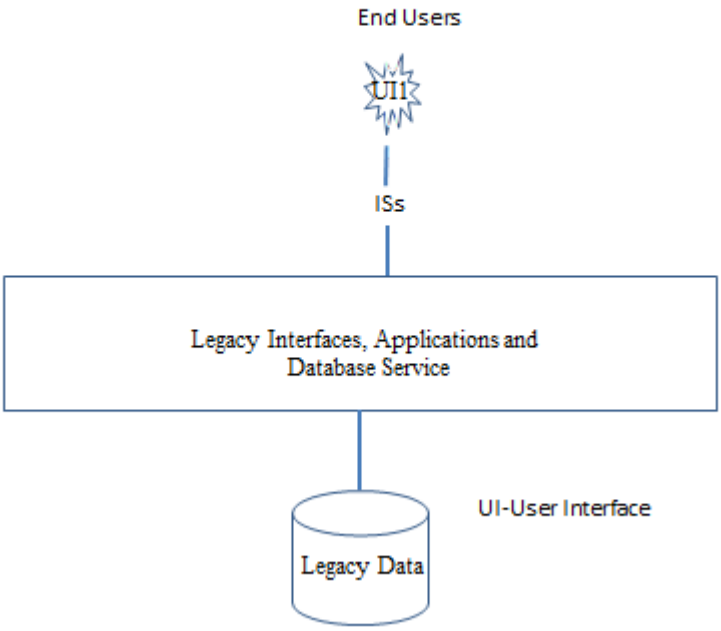


Figure 4-3 Non-Decomposable Legacy IS Architecture[51]

Hybrid legacy Systems

The real business legacy information systems undergoes lots of evolution and changes during their life time. This result in the combination of the three decomposability levels as discussed above.

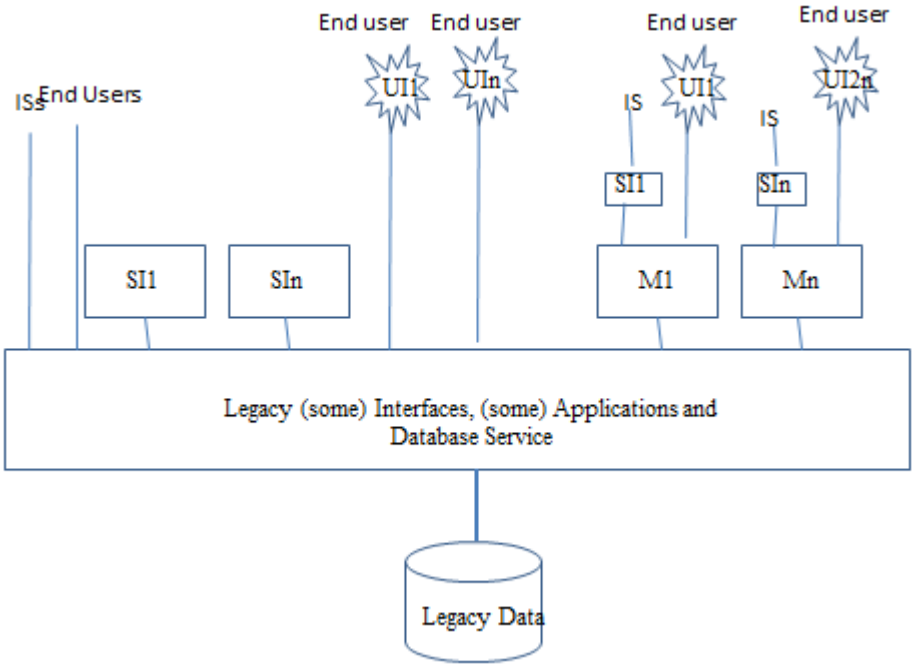


Figure 4-4 Hybrid Legacy IS Architecture [51]

Gateways

The interoperation between the legacy and target systems during migration can be achieved using gateways. Gateways are meant to serve three main functions:

- Insulate some of the components from changes being made to others.
- Translate the requests and data between the mediated components.
- Coordination for consistency between legacy and target data.

The gateway placements depend on the levels of decomposability.

- In case of decomposable systems a gateway would be placed between database services and application modules. This is called “database gateway”.
- In case of semi decomposable systems, it would be placed between interfaces and the other information system components. This is called “application gateway”.
- The non-decomposable systems have a black box representation with “information system gateway”.

4.1.2. Steps involved in Chicken Little Migration

The following steps are involved in the migration process using Chicken Little strategy:

- Install the target environment iteratively.
- Analyze the legacy IS iteratively.

- Decompose the legacy IS Structure iteratively.
- Design the target applications and interfaces iteratively.
- Design the target database iteratively.
- Create and install the database gateway iteratively.
- Migrate the legacy database iteratively.
- Migrate the legacy applications iteratively.
- Migrate the legacy interfaces iteratively.
- Cut over the target IS iteratively.

4.1.3. Advantages

- For each incremental migration, it's possible to track the ROI on the project and the cost involved. So, issues can be identified at a very early stage.

- The target information system is in operation and at the end of each migration increment. So, less failure risks are involved.

- The legacy system is shut down only when the target IS is fully functional. The migration does not result in service downtime.

- Cut-over complexity is less as on each increment only parts of the legacy system will involve risk.

4.1.4. Disadvantages

- Migration of semi decomposable legacy information systems is complex. Migration of non-decomposable is more complex.

- Size of the increment as well as the order of the increment is not defined. For some migrations, it might take years to complete. Also, the changes during those years need to be taken care in both the source and target environments.

- The usage of Gateways introduces serious limitations. First, they are difficult to build and operate.

- The gateways do not offer any support for managing transactions. There is no way to manage consistency between legacy and target systems.

4.2. Cold Turkey Migration Strategy

This migration strategy is otherwise called as butterfly strategy. This strategy is based on the principle of "all at once" i.e. migrating all the legacy components in a single step. This

approach can be otherwise referred as Big Bang approach. This approach involves rewriting a legacy IS from scratch to produce the target IS using modern software and hardware techniques on the target environment [52].

4.2.1. Principle involved

In this methodology, the legacy system is completely rewritten from scratch based on modern software and hardware technologies. This leads to complete re-engineering of the legacy system [52].

4.2.2. Steps involved in Cold turkey migration strategy

Ganti and Brayman [56] have mentioned guidelines about migrating a legacy system to a distributed environment. They proposed that the business is first examined and reengineering of business process is carried out. The legacy information systems are attached to these processes. It's then analyzed to determine the systems with data and business logic of value in the proposed target environment. Consequently, a set of processes are analyzed along with the associated legacy systems. New applications are developed and deployed on the target environment. The method emphasizes that the migration should cause little to no downtime. However, the criteria to cut-over to the target environment are not mentioned [52].

The MILESTONE project [55] developed the butterfly methodology. The methodology emphasizes that data of a legacy system is logically the most important part of the system. It proposes that there should be a clear separation between the target system build and data migration phases. The legacy data store is frozen just before the data migration process begins. Once the migration begins, the legacy data store become read-only store. The manipulations that need to happen on the legacy data during the read-only phase are redirected to some auxiliary data stores (TempStores) via a Data-Transformer and Data-Access-Allocator. After all the original legacy data has been migrated, the TempStores are frozen, while a new TempStore is created to store subsequent new manipulations.

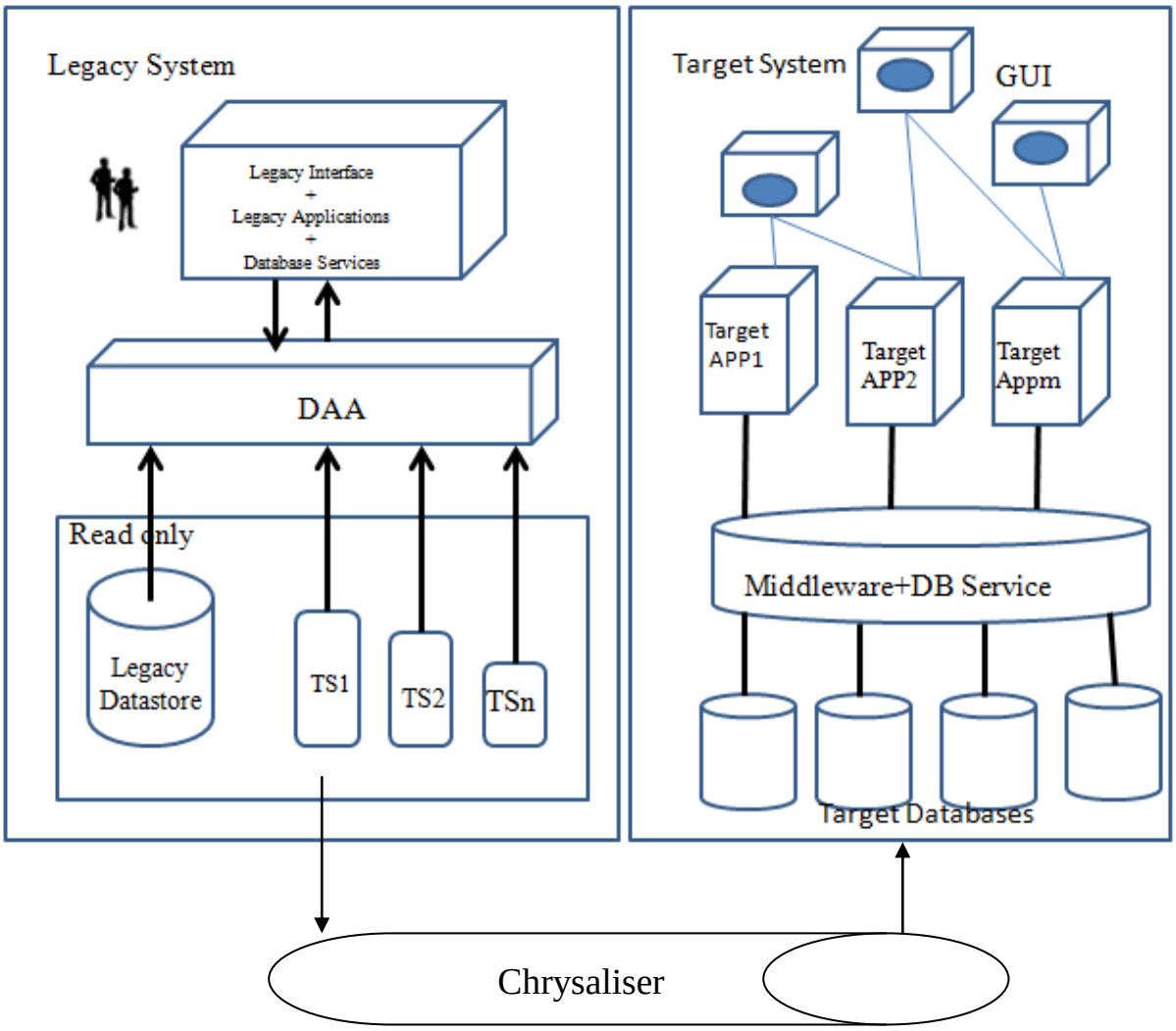


Figure 4-5 Butterfly migration process [55]

Chrysaliser – It's used to transform the legacy data to the target system.

4.2.3. Advantages

- If the legacy database is simple and if the businesses do not change too fast, this method could be used [52].
- It can be very efficient as there is no auxiliary activities like building a gateway, planning and co-ordination with the legacy system. The target system is built and migration happens at once.

4.2.4. Disadvantages

- A better system must be promised-It is nearly impossible to propose a one-for-one rewrite of a complex IS [53].

- Specifications rarely exist.
- Undocumented dependencies frequently exist.[51]
- Legacy ISs can be too big to cut over-There are no techniques to migrate the live data from the legacy IS to the new IS within the time that the business can support being without its mission critical IS [51].
- Management of large projects is hard.
- Large projects tend to bloat. The re-developed systems might not meet the requirements of organizations as businesses evolve [52]
- Failure risk is too big [52]

4.3. Enterprise Knowledge Discovery- Change Management Method (EKD-CMM)

4.3.1. Principle involved

Enterprise Knowledge Discovery - Change Management Method(EKD-CMM) is a method to identify an organization's objectives, business processes and the information systems supporting it and helping it to develop schemes to transform its businesses [4].

Enterprises are complex systems with multiple actors, business processes and technologies. EKD-CMM framework helps in understanding how the enterprise functions currently, reasons for change, alternatives available to realize the change and how to evaluate these alternatives.

4.3.2. Steps involved in EKD-CMM Method

EKD-CMM satisfies two requirements

1. Assisting Enterprise knowledge modeling using EKD.
2. Guiding the Change Process using a road-map

Enterprise Knowledge Discovery (EKD)

Enterprise Knowledge Discovery mechanism results in the generation of enterprise models. As shown in these enterprise models represent the various views of the enterprise. The various views are for example, the goals of the enterprise, impact of these goals on business processes, the constraints on these processes, the information system and the interfaces needed to realize these functionalities.

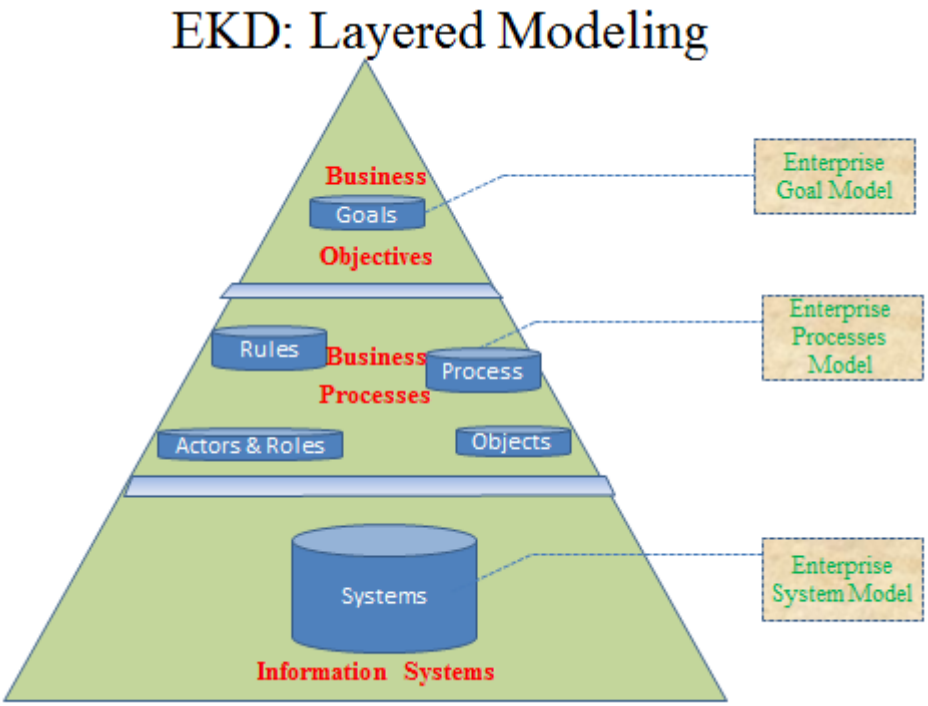


Figure 4-6 Layers of Modeling

Its defined in detail in Appendix A.

Change Management within EKD-CMM:

Change management in EKD-CMM is based on the premise that business change is goal-driven. The need for business change is externalized in terms of business goals.

The need for change is typically stated in a simple manner as a change vision.

"A classic example is John F. Kennedy´s statement: '*to send a man to the moon before the end of the decade*'. Thus, the change process is the process of transforming the vision into a new model."[4].

Within the vision, many habits (legacies) exist. Some are based on formally stated goals, policies or competing visions. Others are just regularly observable phenomena for which no predefined structure or reasons are known a priori.

The task inherently consists of 2 steps. First, relevant habits must be analyzed and the goals, policies and the visions must be made explicit. This defines the existing practices and leads to "As-is" Model as it describes the "As-is" state of the system.

Second, the new vision is modeled and describes the "To-be" model.

The change from "As-is" to "To-be" model is described as a change-process model. It's a complex model which is further divided into 2 steps, modeling the alternative scenario for

change within a change process model and secondly, selecting the appropriate scenario for change.

Thus EKD-CMM modeling involves 3 models

1. As-is Model – This defines the current processes and goals

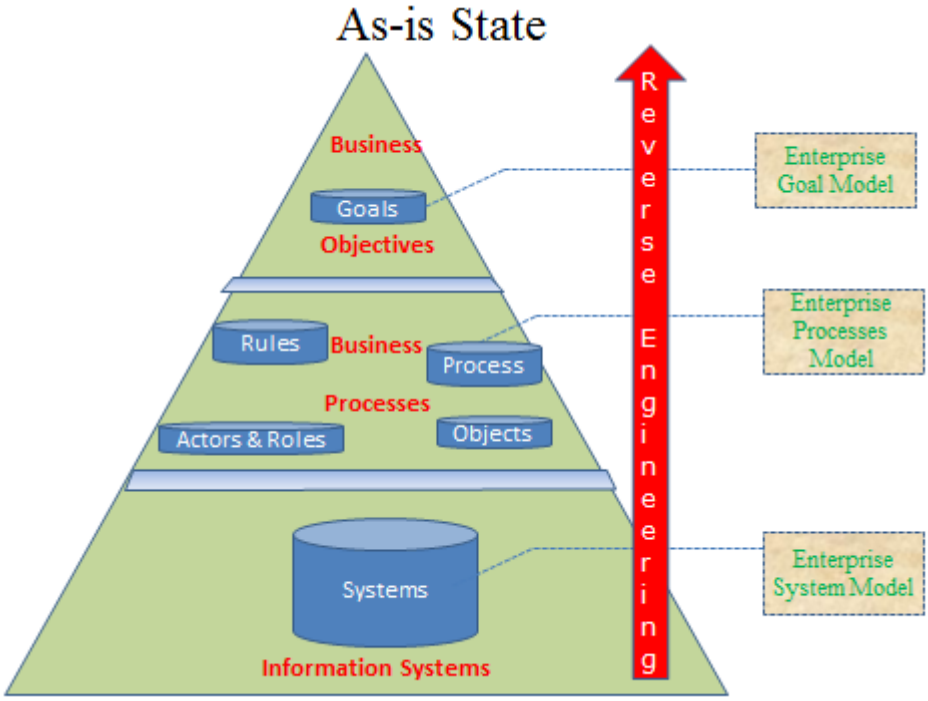


Figure 4-7 As-is State

2. To-Be Model - This defines the future business processes and goals

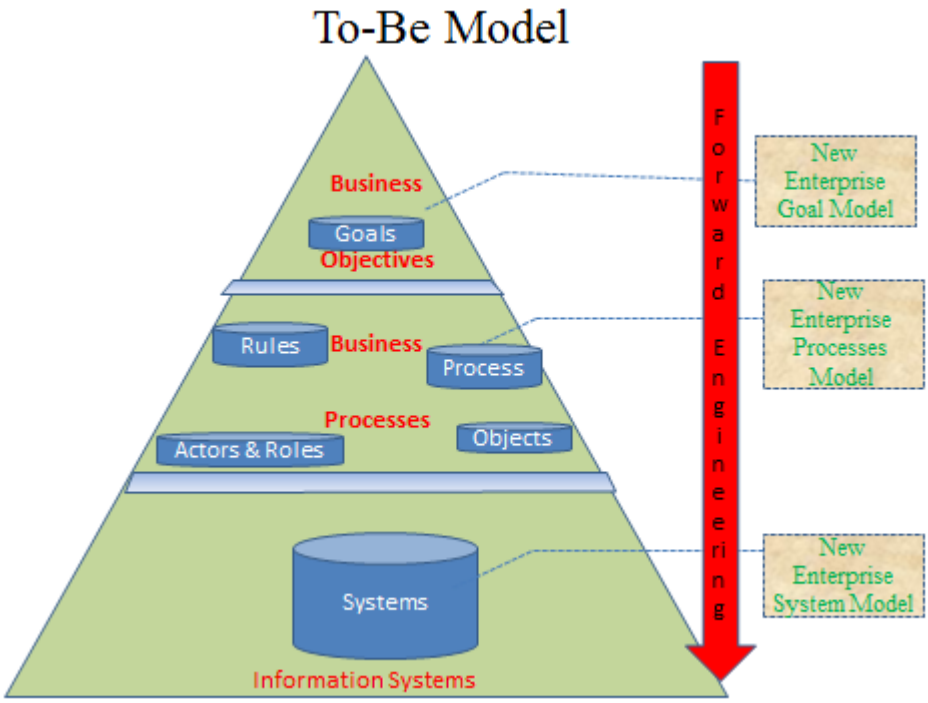


Figure 4-8 To-be Model

And

3. Change process model: This model consists of 2 steps
    1. Defining the road-map as was shown in Figure 4-9 Road-Map which lists the possible routes to realize the "to-be" model
    2. Selection of an appropriate route among the possible routes.

The road-map (like the physical road-map) provides different routes to reach a particular destination. Choice of one route depends on one's intention (goal, business processes) and availability of the vehicle (possible business process or information systems). As the goals change, dynamic selection of a route is supported by the EKD method.

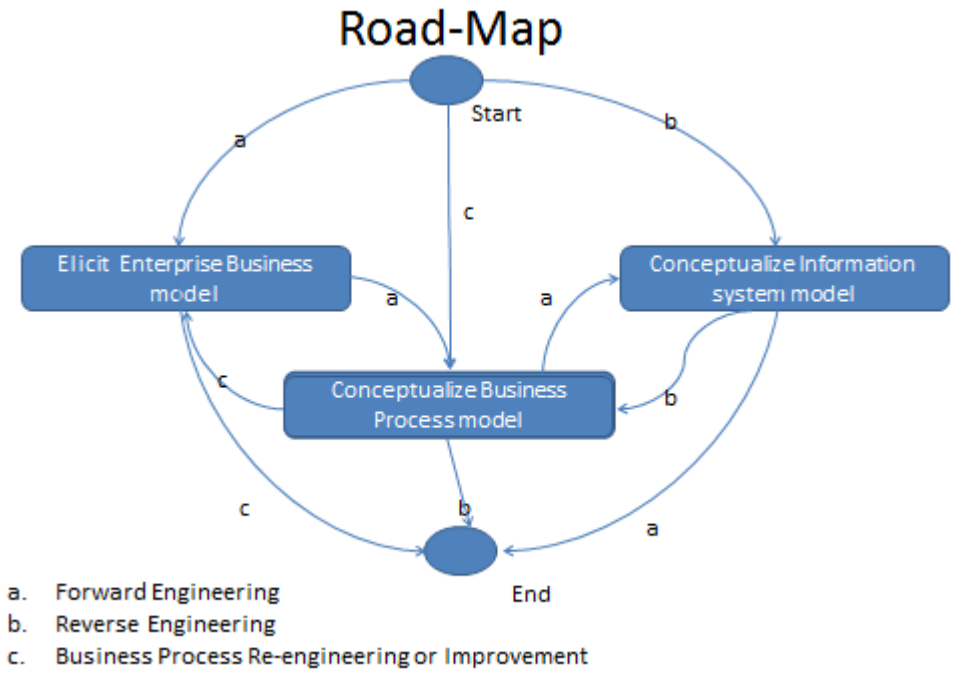


Figure 4-9 Road-Map

The figure shows 3 example routes in the EKD road map.

“a” is the route that will be followed if one wants to perform Forward Engineering “b” is the route for reverse engineering and

“c” is the route for business process re-engineering or improvement.

Despite the increased understanding of the problem domain among the stakeholders, improved communication and collaboration among the participants, EKD is very resource intensive [20].

4.3.3. Advantages

- Provides multiple views of the enterprise which will be helpful for strategic migration efforts.
- All the conflicting goals and constraints and the functional specification of the “to-be” system are identified and externalized in the form of process models. At the end, a “to-be” architecture model is given as a requirement to the IT.

4.3.4. Disadvantages

- Modeling is expensive. It requires sufficient time and resources for the modeling.

- No automatic transformation of the process models from “as-is” state to corresponding process models in “to-be” state is possible.

4.4. Service Oriented Architecture

4.4.1. Principle Involved

Linthicum [19] proposes that Service Oriented Architecture(SOA) is an architecture pattern that is about viewing the IT architecture in terms of services. It allows us to address a system as a set of services and abstract those services into a single domain where they are formed into solutions. With the evolution of WS-* technologies, SOA can be used to bring in agility into an enterprise where business drives IT.

Linthicum [19] further describes SOA as a strategic framework of technology that allows loosely-coupled services with well-defined interfaces that provide business functionality. These services can be shared or reused across and beyond the enterprise. These services can be discovered through a registry/repository or other directory, and can be assembled and dis-assembled to meet current business process demands. Thus SOA provides agility aspect to architecture, allowing us to deal with system changes using a configuration layer rather than constantly having to re-develop these systems.

4.4.2. Steps involved in SOA

- The functionality of the system is exposed as a set of loosely coupled services.
- The services are registered and discovered in a registry/repository.
- The services are composed into business processes that satisfied the business requirements of the information system.

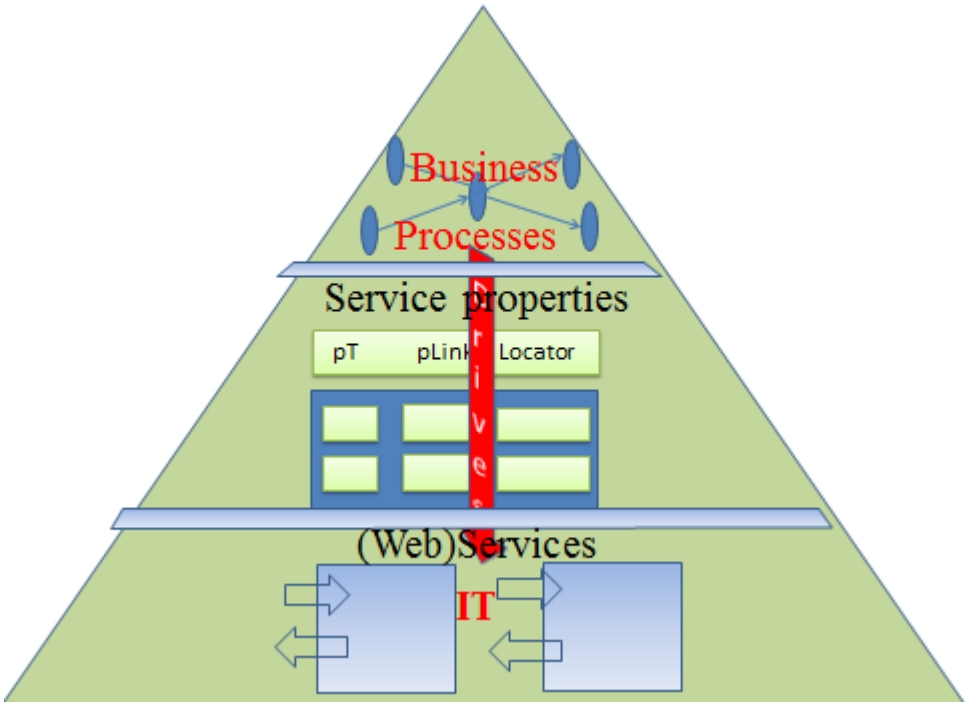


Figure 4-10 Service Oriented Architecture

4.4.3. Advantages

1. Reuse of services: Leverage application behavior from application to application without significant amount of recoding or integration. One can access same application functionality without having to port the code, leveraging remote application as if it existed locally.

2. Agility – one can change the business processes much faster to keep pace with the changing business. Changing a business process is much faster than changing code with respect to changing business.

3. Monitoring – One can monitor information, service at different levels of granularity in real-time. Thus allows us to respond in an agile manner.

4.4.4. Disadvantages

1. Service Governance is difficult to implement and maintain.

2. Identification of business processes and rules are difficult by doing code analysis. This is needed to identify and define services from the legacy code.

3. This needs to be driven from the business to the IT. This is often difficult, as showing ROI on investing in a SOA project in the short-term is difficult.

4.5. Architecture Driven Modernization

Architecture driven modernization can be defined as the process of understanding and evolving software assets for the purpose of migration [45]. It's based on an emerging set of OMG standards.

Modernization bridges the gap between existing architectures and strategic architectures.

4.5.1. Principle Involved

Khusidman et al. [47] proposes that modernization of Information system can be viewed from within 2 domains in 3 different perspectives.

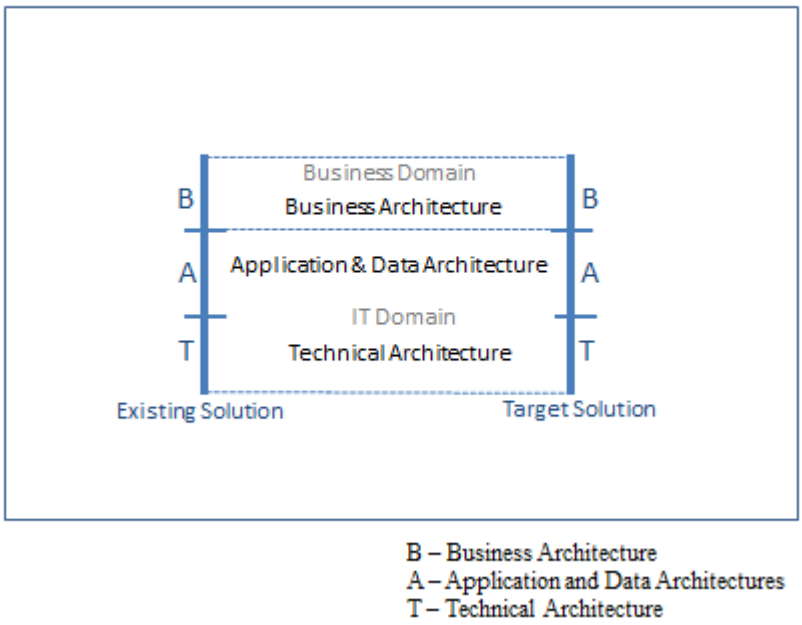


Figure 4-11 Business Vs IT Architecture Domains

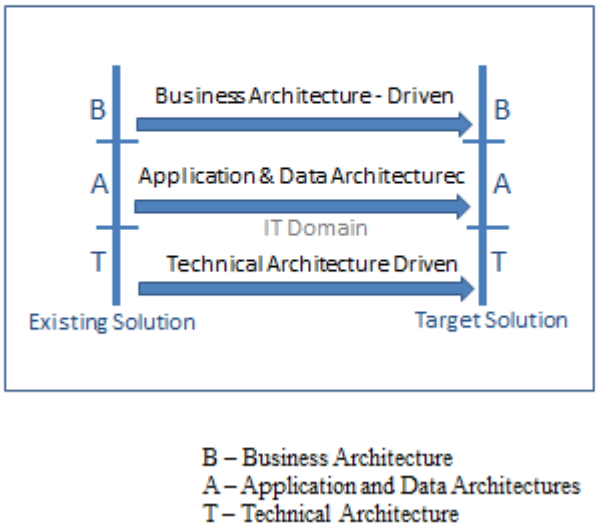
Figure 4-11 depicts the business domain, which is comprised of the business architecture. It also depicts the application and data architectures and technical architectures in the IT domain.

The business domain consists of models and corresponding diagrammatic views of operational governance, business semantics, business rules and business processes. The IT is comprised of application and data architectures as well as technical architectures.

The existing solution is on the left of figure, while the target solution is shown on the right side of Figure 4-11. The target solution keeps changing as more parts of the existing solution are migrated.

4.5.2. Types of Software Modernization

Khusidman et al. [47] proposes that the software modernization effort can be classified into 3 categories viz. 1.business architecture driven by the business, while application/data architecture driven and technical architecture migration are driven by the IT.



B – Business Architecture
A – Application and Data Architectures
T – Technical Architecture

Figure 4-12 Modernization Drivers & Trajectories

Khusidman et al. [47] also mentions that Modernization projects typically focused on transforming technical architectures. However, as the organizations understand the value of transforming higher level architecture, formal interoperability across these domains become more critical. As enterprises look to transform themselves, modernization needs to across all the architectural perspectives. A horse-shoe model is used to explain the interoperability concepts.

4.5.3. Steps involved in Architecture Driven Modernization

The steps involved in ADM represent a horse-shoe and hence it's also referred as ADM Horseshoe Model.

ADM usually involves one or more components of the IT architecture. Each component has its own trajectory to migrate from its "as-is" state to the "to-be" state as shown in Figure 4-13 ADM Horseshoe Model.

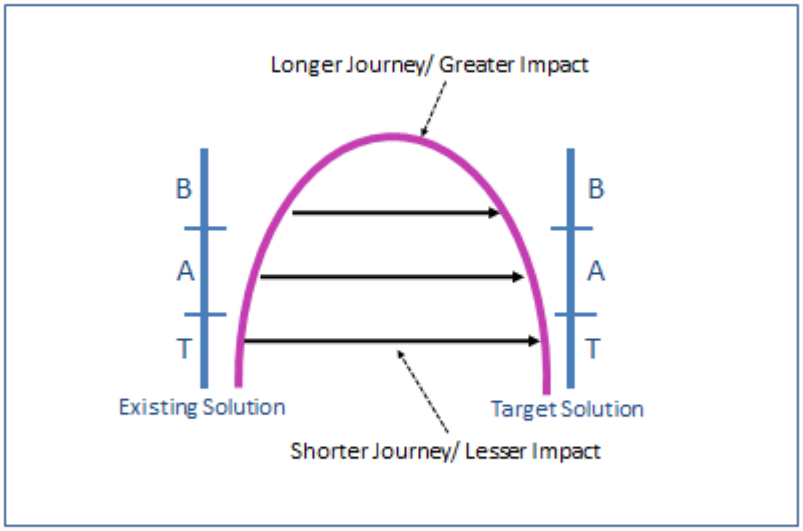


Figure 4-13 ADM Horseshoe Model

There are 3 elements that are common to every transformational path, regardless of the level of architectural impact. They are

1. Knowledge discovery on the "as-is" state. This can occur with different levels of abstraction as the scope of the projects involved.

2. Target architecture definition. Here, analysts create a framework for the target solution. The existing solutions are mapped / transformed within this framework.

3. Transformative steps that move the "as-is" state to the "to-be" state. The approach ranges from the physical (e.g. language migration) to something more abstract (e.g. business process mapping)

These transformational paths must be synchronized both vertically (business-to-physical implementation) and horizontally (existing-to-target). For an enterprise, multiple collections of these journeys might be needed based on business and IT requirements.

In the next sections, we define the ADM Horseshoe model for the 3 modernization types.

Technical Architecture Driven Modernization

It's historically the most commonly applied type of modernization project. These projects are motivated by risk due to platform or language obsolescence, cost of ownership, system usability or other issues that can be addressed through a physical change.

Modernization projects of this type can be a project that involves, moving from one platform to another, from one language to another or UI replacement.

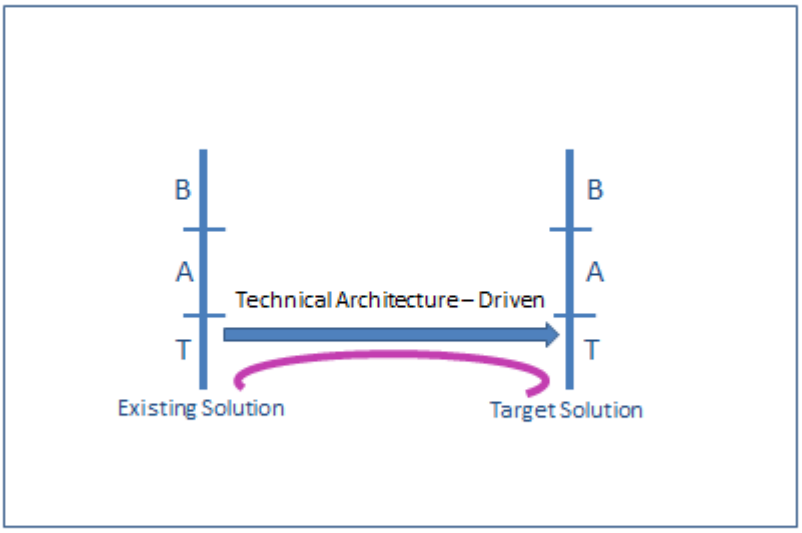
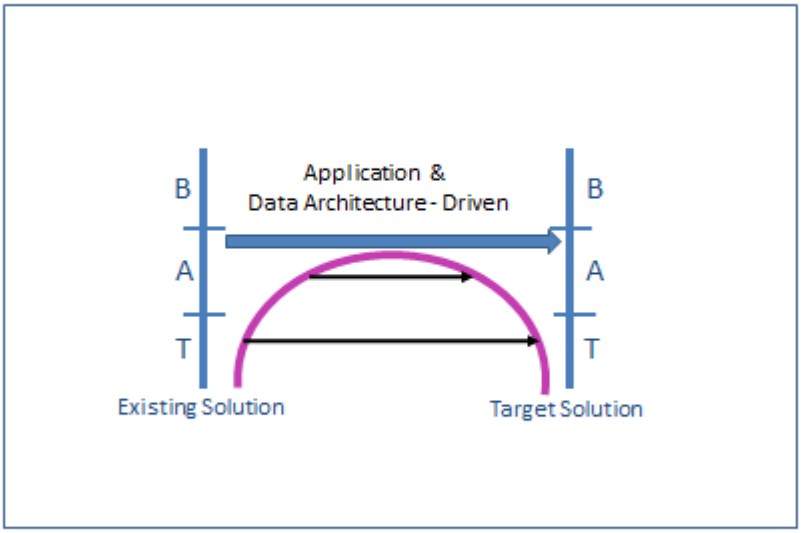


Figure 4-14 Technical Architecture Modernization

There is a fine line between projects executed at the technical architecture level and a project executed at the application/data architecture level. When there is an impact on program design, system-level or data design factors, it is said to be application and data architecture driven. For e.g. a project that involves moving from an object oriented language to a procedural language.

Application/Data Architecture Driven Modernization

This modernization moves up the scale of modernization paradigm. A project to abstract, redesign and redeploy existing applications in a model driven architecture (MDA) falls into this category.



B – Business Architecture
A – Application and Data Architectures
T – Technical Architecture

Figure 4-15 Application/Data Architecture Driven Modernization

These sorts of projects are motivated by a number of factors. For e.g. application architecture modernization project might involve multiple applications that no longer serve the needs of the business through incremental activity. It may alternatively involve a data architecture that is out of sync with strategic information requirements.

Another e.g. could be a project where a set of applications are redesigned and transformed into a common application that uses a redesigned data model with a platform migration. This project goes through both technical as well as application/data architecture trajectory to reach to the "to-be" state.

Projects like these have been executed by IT from time to time, but they do not apply the formal disciplines required in a modernization project. Also, the inter-operability standards with MDA and other standards are still evolving. However, the IT architecture is still not aligned to the business architecture.

Business-architecture Driven Modernization

Business architecture modernization incorporates business architecture models, application and data architectures and technical architectures to migrate to the "to-be" state.

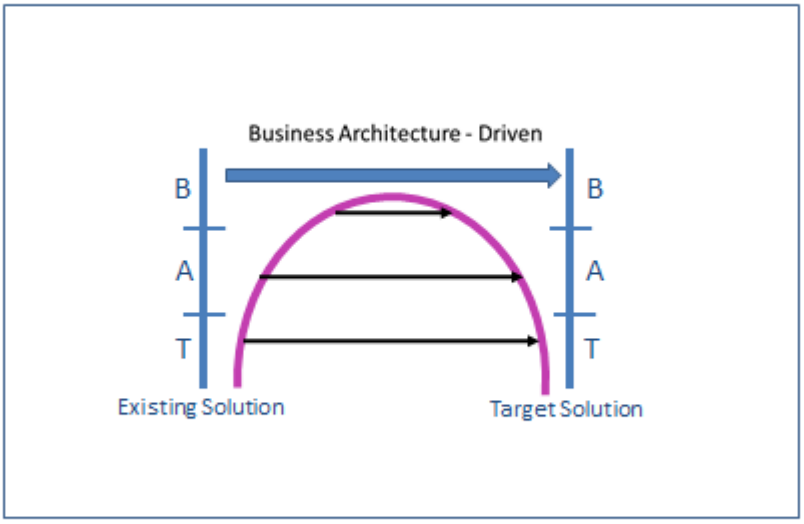


Figure 4-16 Business/IT Architecture Transformation

This modernization projects align application and data architecture models with the business models. These business models include a combination of business rules, business semantics, governance and business processes.

4.5.4. Advantages

- ADM provides processes for assessing existing architecture as well as for capturing, reusing and migrating existing software artifacts. It also provides processes for analyzing, designing, building and deploying target architectures.

- ADM also provides software standards that facilitate the modernization and development process.

4.5.5. Disadvantages

No formal mappings between ADM and MDA models, the transformation standards to support these mappings, formal mappings between ADM Model and the spectrum of business models, and the inter-operability across business models.

4.6. Summary

4.6.1. Applicability of these strategies to Cloud migration:

Enterprise application portfolio consists of different kinds of applications developed over different periods of time. Each of these applications needs to be handled differently as their requirements might require them to be. Our analysis of the different cloud migration case studies reveal to us that there is no one strategy that fits all the requirements of all the applications in an enterprise. Below, we show the applicability of these strategies with respect to cloud migration.

Chicken Little Strategy

- For e.g. Darwin framework is an integrated automation framework intended to dramatically reduce both the cost and risk of migrating workload. It is based on Chicken Little strategy and Ward et al. [48] have extended this framework to migrate workloads into clouds.

- However, Bisbal et al. [53] have concluded that the usage of gateways brings serious limitations to this strategy. They pointed out that the gateways offer no transactional support which implies there is no way to ensure data consistency between the legacy and target systems.

- Further, they point out that the gateways offer no way to homogenize the structural and representational differences between the two databases and that they are difficult to build and operate.

- The size and the order of the increments are not defined by this strategy. Also, the framework does not offer any guidance for decomposition when it comes to black box or spaghetti-like information systems. Hence, this strategy might not be applicable to migrate such information systems.

- Despite these disadvantages, it is a very good strategy to migrate modular and semi-decomposable information systems as one can track the progress of the migration after every increment. The risks involved are very less and the migration can be completed with zero downtime.

Cold Turkey

- Though cold turkey is a risky migration strategy to migrate mission critical information systems, it has been used successfully in cloud migrations.

- For e.g. when the National Oceanic and Atmospheric Administration of the USA has migrated to a cloud based e-mail adoption, they had to use this strategy to migrate 23,000 email calendars at once as the employees calendars are deeply integrated and had to be moved all at once [85].

- Similarly, for an enterprise migrating from a UNIX based environment to Linux, Cold turkey might be used as has been suggested by Gartner [86].Migrating using this strategy might be the most cost and time-efficient way though the risks involved are usually high.

Enterprise Knowledge Discovery – Change management framework.

- Though this framework helps in anchoring the change across the different stakeholders in the enterprise, the modeling is not standardized. Hence, Model transformation standards do not exist. Hence it might be better off to use other MDE strategies that are standardized that support automatic model transformation.

Service Oriented Architecture

- It is seen as the de-facto strategy for most of the successful cloud computing projects. Since cloud is delivered as a "service", architecting an information system in Service Oriented Architecture style provides great flexibility, agility and loose coupling. Most of the cloud computing pioneers have used SOA as the strategy to successfully leverage cloud resources [87], [10].

- However, migrating legacy system to service oriented architecture with reusable services might result in performance loss and considerable effort and resources.

Architecture Driven Modernization

- Architecture Driven Modernization strategy has been adopted as the strategy for the recent EU research project REMICS [61] which is about migrating legacy applications to service cloud paradigm.

- This Model-driven architecture strategy has been supported by OMG and the hence models described are being standardized to support (semi)automatic model-transformations among the different models involved.

- Models like Knowledge Discovery Meta-Model[74] have been standardized and tools like MoDisco[59] have been developed to validate the ideas behind the strategy. However, still many of the modeling standards are either in their first versions or not defined at all. They need to be defined and supporting tools are needed.

All the above strategies have their scope for migrating applications to a cloud environment as shown above. Since, our process model is a generic process model that can be used to migrate all types of applications to cloud, they all are relevant and can be used in conjunction with our process model.

5. Resulting Cloud migration process model

The resulting migration solution is an abstract, vendor-independent, model-driven process model. We use layered modelling to define the "as-is" model, the "to-be" model and the change process model. As defined in 4.3.2, the change-process model uses a road-map to complete the change-process.

Essentially, we are doing

1. Reverse Engineering to define "As-is" model
2. Forward Engineering to define "To-Be" model
3. Business process Re-engineering to realize "To-Be" model.

At the end of the forward engineering step, we design a new enterprise system model.

This enterprise system model is essentially a list of requirements. Using these requirements, an Information system suitable for an Enterprise's vision is built in the change-process model.

It must be clarified that the idea of model-driven engineering is used here. So, one can also use a similar framework like [35] for the development of information system. The model driven engineering helps in identifying conflicts from different views of the enterprise, which can then be overcome to provide the requirements to the IT. It might be possible to (semi)automate code from these models. It's not possible to completely automate code generation from the models as the models are used to represent the "As-is" and "to-be" state in detail. For e.g. if one models the business rules using Semantic of Business Vocabulary and Business Rules(SBVR) , code generation is only possible at certain cases [29].

We are going to use Service Oriented Architecture (SOA) as a strategy to reverse the Information system in the "As-Is" model to perform cloud migration. [6] Cloud is offered as a service much like water, gas, electricity and telephony [23]. The corresponding IT-system needs to take advantage of it and should be built in Service Oriented Architecture (SOA) style, where the systems are loosely coupled. Hence, where-ever possible, the component to be migrated is encouraged to be built SOA style.

In a cloud scenario, governance is one of the most important aspects. It might be the case that the platform might be updated by the cloud provider without the control of the enterprise. Hence, it's vital, to have a loosely coupled architecture. Also, with cloud, agility is very much a reality. But in order to be truly agile, software needs to be composed as a service because

that is the only way we can have most software re-usability [23]. It also allows us to compose applications in an agile manner. Additionally, one can change the IT system on a configuration level without recoding services as the business changes. Thus, SOA is emphasized as a strategy to realize cloud computing for an enterprise [22, 23 and 24].

Figure 5-1 Cloud Migration process

5.1. Cloud Migration sub-process

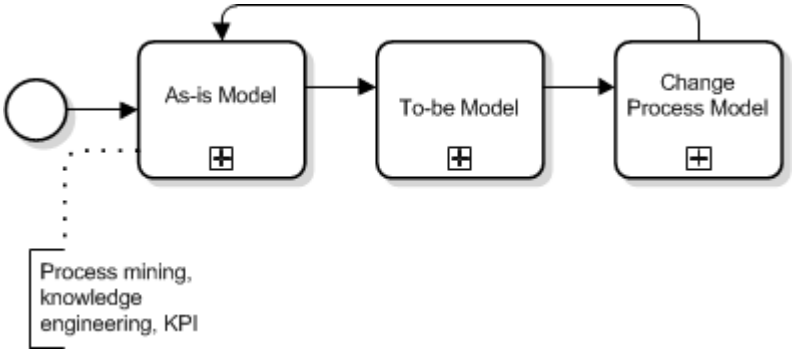


Figure 5-2 Cloud Migration sub-process

This sub-process follows EKD CMM which was explained in 4.3 to model cloud migration [2].

As-is model: This sub-process is focussed on reverse engineering the enterprise to represent the current state of the enterprise, viz. the information system, the business rules and the current business processes with the actors involved [4]. We also define/measure a Key-Performance Indicator (KPI) Metrics which organizes our priorities from the topmost to the least significant of our enterprise. During each cycle, the current state w.r.t the KPI of the previous cycle is measured.

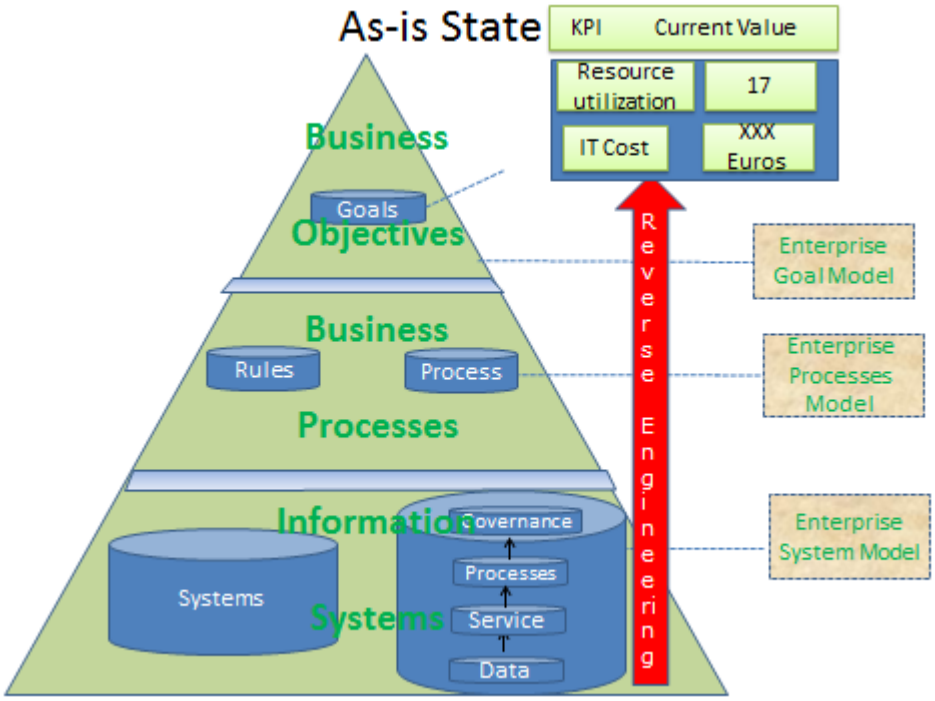


Figure 5-3 “As-is” Model

To-be Model: Armed with the knowledge of value of the current state in the enterprises and the KPI Metrics, this sub-process is modelled. Here again, the multiple views of the enterprise are modelled. The various aspects like security requirements, compliance requirements are modelled in the rules model. Business Process Improvement or Re-engineering is performed and a new business process model is designed. Then a test plan is modelled by IT and business that defines the functional and non-functional requirements of the new architecture. Roles are updated for the actors involved. IT makes use of these models, to design the “To-Be” state of the architecture. This “to-be” state is for this cycle of migration for the selected problem domain. In each cycle, the new enterprise vision is updated; the new business processes that are needed to achieve the required KPI are modelled and the same process is repeated [4].

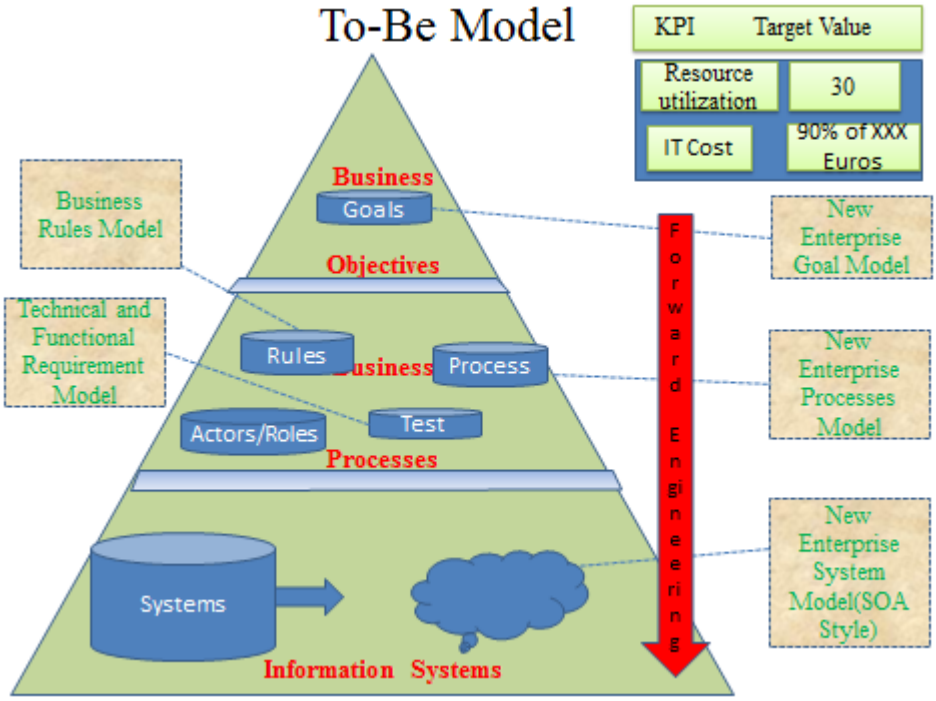


Figure 5-4 To-Be Model

Change-Process Model: In this sub-process, the actual migration steps are carried out. It is here that the value of the collaborative nature of the EKD way of working comes to the fore [4].

With all the issues, problems, constraints and the "to-be" architecture modelled collaboratively, the task of the IT can be that much more focussed knowing fully well the requirements and the KPI for the cloud migration.

For each candidate that was assigned as a candidate for cloud migration in the "to-be model" sub-process, many different routes are identified to achieve the target KPI. The components are re-developed/tweaked for each of the target cloud option and a pilot is executed. Then, a target platform that is more consistent with our defined KPI's is chosen. All the stakeholders involved collaborate to identify whether the target platform is consistent with the functional and non-functional constraints for the component as well as with the target KPI of the enterprise. At the end of this sub-process, a target platform is identified for each of the components.

Then, the components are deployed on the selected target platform. It is then continuously monitored to ensure services are provided with an appropriate level of quality of surprise as well to ensure availability and disaster recovery.

Next, periodically, the current "as-is" state of the system is evaluated to ensure that the enabled cloud service is aligned with the defined target KPI. It might be the case that, a target platform might have been very attractive w.r.t defined KPI [8]. However, over different

cycles, it might be the case that the initial savings might not be the same as the cloud service costs might vary. It might be the case the cloud service might face disruption resulting in business loss as well as regulatory fines, etc. [9]. Since cloud services are most often shared (in private cloud with other IT departments or in public, community cloud with other organizations) disruption might occur that are no-fault of one's own. These events can not be taken into account while choosing the target platform. Hence we propose cloud migration as not a project, but rather a continuous journey. It might not be optimal to choose a target platform and monitor the service in the target platform alone. It might be so that other candidate platforms or new platforms are better in line with the target KPIs [10]. If the services are carefully designed, one can port across different platforms with changes in configuration depending on the required quality of service. Or one service might need to be deployed across multiple clouds to ensure availability or to handle spill-over traffic. Hence, this link to the "as-is" that indicates cloud migration as a continuous journey is critical for cloud migration. Smith [28] describes this feedback loop reflects the dynamic nature of service oriented environments.

Thus we approach cloud migration as a continuous journey where we learn from each cycle of cloud migration. It's from this context, that cloud migration should be performed top-down. If it's performed bottom-up, then one department might be able to learn from its own mistakes, but it's difficult to learn from others. When driven from top-down, the issues with cloud services and cloud service providers are more anchored in the organization and that will act as a knowledge base to better prepare the next cycle of cloud migration. Also in a top-down approach, the process starts with the overall goals of the project. When the goals change, the processes change and that triggers changes in services. Schepers et al. [31] proposes that this requires more coordination effort and may cause resistance to change, yet it leads to a more coherent solution.

Organizations can develop strategies to re-create those non-functional issues and test for resilience of the next set of components. Governance policies need to be developed and evolved over a period of time As an enterprise migrating to cloud, it's important to evaluate the "as-is" state of the migrated components and modify the KPIs and use those experiences to better compose the services in the "to-be" system. A well-defined system should be resilient to handle failures from a cloud service provider.

Also, in order to bridge the gap between IT and business, this needs to be driven from the business to the IT as IT can not be expected to guess the direction of the business. More importantly, the focus of the IT is most often on performance, availability or technical aspect of the business rather than on the business itself.

5.2. As-is Model

This “as-is” model of the enterprise does not try to include the entire enterprise’s applications. A problem domain is chosen. The problem domain should not be too big a domain set. It should be big enough to have a meaningful effect, yet, small enough to be completed in a sufficiently short frame of time. If the domain is too big, then organization and political obstacles might become a bottleneck in one’s cloud journey [19].

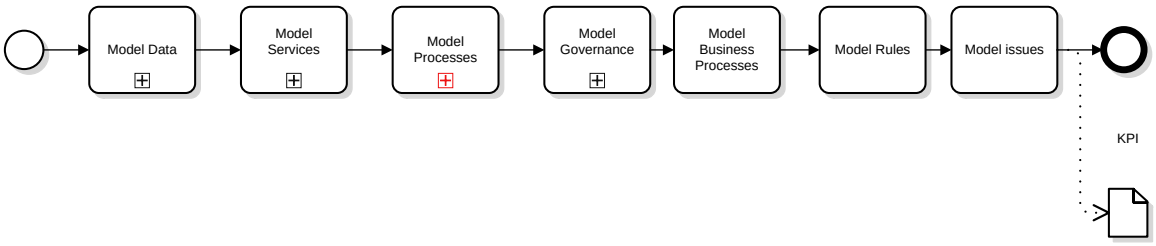


Figure 5-5 As-is Model

This sub-process can be viewed as a bottom-up approach to identify the current state of the enterprise. Here, we do reverse-engineering to identify the current state of the enterprise.

It might be the case that an enterprise might have to do process-mining to externalize the business processes from the system. Some processes might need to be identified not only from the different systems involved in the enterprise but also from the persons involved in the company where the processes are in the form of knowledge within the people working at the enterprise. Essentially, we are building a triangle bottom-up in Figure 5-3 “As-is” Model,

1. starting with data, services at the bottom
2. identitfying business processes
3. identifying corresponding business goals from these processes

While modelling as-is sub process, we go from data to services to processes to security and governance. It’s modelled in this fashion as in most IT architecture with legacy systems, these are not externalized.

It’s important to map out the processes from the IT architecture from the most-granular element and then abstract further up. There are multiple reasons for moving along in this direction:

1. Good architecture foundation of the existing “as-is” state.

2. Moving from data to the services to the processes and applications is a great way to move from the most primitive form of IT to the most complex.

3. Doing it from data enables one to identify redundant data. It also enables us to identify and categorize data based on their sensitivity, in terms of privacy and security. Too often, this is ignored by IT which might be relevant in a cloud scenario.

4. Identifying the services enable us to identify redundant services and categorizing critical services. The performance requirements (run-time, access time, write time) and the costs associated with each component (data, service) need to be measured. Essentially, at the end of this knowledge discovery process, the performance and cost of components at different granularity levels are accurately identified. In addition to mapping out the process model for the business process, the functional requirements and the non-functional requirements are identified. This knowledge is used to define the KPI of the as-is state.

Implementation:

Aboulsamh [76] proposes that the business processes in the problem domain are modelled as use-cases based on UML-standard. This allows an enterprise architect to choose a single or a set of business processes to choose from the problem domain. These use-cases can then be used to focus to further model the data, services and the business processes involved in the problem domain. This use case model is used as a reference to drive the migration process.

5.2.1. Model Data

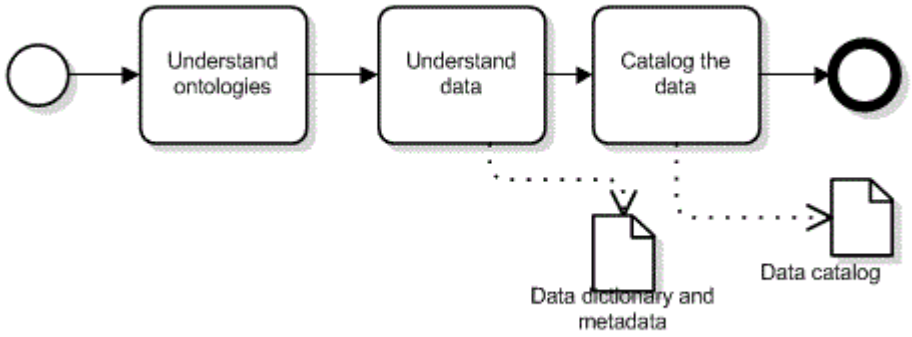


Figure 5-6 Data Sub-process

We are trying to define all relevant meta-data about the applications in the problem domain. We define where the data is now, the data structure, the logical model, the physical model, security issues, and data definition and so on. At the end of this process, a data catalog is designed and a complete metadata layer is formed.

Defining the data catalog might be a challenge as many of the IT systems are legacy systems which are too old or proprietary or both. Any tools and technologies that can help in reverse-engineering existing physical and logical database schemas might be very handy. While those tools help in providing insights into the structure of databases, they can not determine how the information is used within the context of the application or service. We perform the following to identity this information

1. Understanding Ontologies
2. Understanding data
3. Creating the data dictionary

1. Understanding Ontologies

Since the volume of data might be enormous and unmanageable, it might be a good idea to generalize these data into ontologies. This helps us to understand the problem domain better while still keeping it as manageable.

Also Ontologies allow for entity correspondence. Ontologies leverage on data that are distributed across different data sources in an enterprise. For the same entity for e.g. a product, the information such as on-time delivery history, customer complaints and compliments might lie across the enterprise. With ontologies, we can have this entity information in a single location. This also should expose data duplicates which can then be cleaned up [19].

2. Understanding Data

We need to understand where the data exists, gather information about the data (e.g. schema and meta-data information) and apply business principles to identify the information flows. A successful Cloud migration requires a thorough understanding of how data flows through the enterprise and how it is related to core business process and core services. By analysing and understanding the meaning of the data, an enterprise can classify the data from business sensitivity (for e.g. credit card information) to compliance sensitivity (For e.g. medical information) to competitive advantage information. Understanding the semantics of the data enables us to identify proper candidates to migrate to the cloud. In this way, we build the data dictionary and meta-data for each system in the problem domain.

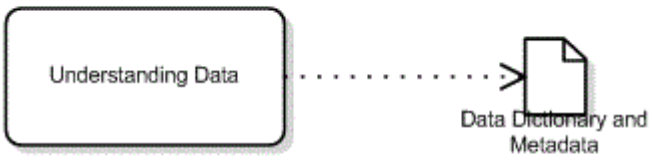


Figure 5-7 Understanding Data

Other Data properties to be considered:

To understand the data, one might first have to identify and list all the data sources, find out who the owners of this data are, relevant technology of the databases, etc. There might be systems like Enterprise Resource planning systems, where the data might be wrapped inside the system. In that case, the ERP application vendor needs to provide the information. There are tools available which can help to build this data dictionary; there are databases that have data dictionary built into the DBMS [19].

It's important to understand the integrity issues that cloud migration might introduce. While some databases come with integrity control features (stored procedures, triggers, etc.), most rely on the application logic to handle integrity issues. With NoSQL paradigms and the fact that the data will be distributed in a cloud environment, integrity issues need to be considered and documented.

Fehling et al. [7] proposes data latency, the measure of how recent the data needs to be, is another property that must be determined. Storage in a cloud offering usually consists of multiple replicas to ensure fault tolerance. This replication also introduces problems/issues that must be considered before migrating to the cloud [7].

Depending on the latency requirements of an application, several issues need to be considered:

For e.g. Fehling et al. [7] describes strict consistency as "A storage offering usually consists of multiple replicas to ensure fault tolerance. It is of major importance that the consistency of the data contained in these replicas is pertained at all times while the performance is of secondary importance." . This might become a challenge in a service/application that requires correct version of the data all the time. The availability might be drastically reduced to make sure that the data is consistent.

Not all services/applications might need strict consistency. Where there is more focus on availability than on consistency, eventual consistency needs to be considered.

Eventual consistency is defined as "a storage offering usually consists of multiple replicas to ensure fault tolerance. It is of major importance that the availability of the data contained in these replicas is increased while the consistency of the data is of secondary importance." [7].

While classifying the data and building the data dictionary, the latency requirements also need to be captured so an appropriate cloud storage solution can be selected.

In this way, we ensure that some services which require data to be updated in a transactional manner are migrated to an appropriate platform that supports these transactional requirements.

For applications that require consistency as well as availability, there have been issues that were identified. These issues need to be considered. For example, Vasilios et al. [62]

summarizes the CAP problem as “in any distributed environment that exist in a special relationship with each other: consistency (whether all parts of the system see the same data at the same time), availability (what percentage of time the system is up and functioning properly) and network partitioning (if the system is tolerant to network failures). His conjecture is that only two out of these three requirements can actually be satisfied at any time by a distributed system.”

Since cloud is a distributed system, this issue applies to cloud and might be an issue for a cloud application. Building the data dictionary capturing all these requirements in detail will help the migration process greatly by avoiding these issues.

3. Building Data catalog

Figure 5-8 Data catalog

Data dictionary was built for each individual service/application. Data cataloging is about combining all the data dictionary of the individual services/applications in a formal representation.

This catalog will be the main artifact of this sub-process. We use this information model to determine whether some data resides in the cloud or on-premise.

Building this data dictionary can become cumbersome for a large enterprise. Especially over multiple iterations of the cloud migration process, this data catalog might contain security information, ownership information, integrity issues, semantic details of the data, sensitivity of the data and so on. While the effort might be enormous, this data catalog will become the central repository for identifying new business flows and to understand existing flows. Since there are no standards for data catalog in a cloud scenario, the more detailed the information, the better the data catalog is. Maintaining this data catalog up-to-date is very crucial and needs to be taken care of.

Implementation:

The enterprise knowledge discovery process is applied on the problem domain. For this knowledge discovery, a tool like MoDisco [59] can be used to model this in Knowledge discovery meta-model format [60]. The data about the software system and its environment can be obtained and used to build the data catalog of the problem domain. A generic configuration management system [77] and specialized migration oriented utilities [78] can be used to obtain both static (IP, OS type, etc.) and dynamic configurations are accumulated.

5.2.2. Model Services

Service is exposed out of an application to access both the behaviour and the data using a standard interface. Even legacy systems functionality can be wrapped with convivial software known as "frontware", thus enabling web access to its functionality [21]. In the case of tightly coupled systems, the use-case

By tapping into the behaviour of services, enterprise can combine services into a composite application. This approach provides flexibility for the architecture. With Cloud resources, an enterprise now has the ability to compose applications quickly as the businesses change [23].

While identifying the services in the enterprise knowledge discovery, these services might be classified as applications that can be run in-cloud or on-premise and services that can expose their services to applications on the cloud and in-house. This is an important step in the knowledge discovery process. We identify and define the services, whether it can be run in-house or on the cloud and are accessible from both the systems [19]. The services are also grouped together based on the business process it supports under process domains [26]. It might also need to be grouped under "product domain" (all services related to a particular product) or "geographical domain" (useful to be in consistent with compliance/legal issues) or functional services (a specific department e.g. sales, accounting) or "technical domain" (for e.g. built using COBOL, C++). Also, Smith [28] points out that Service identification might still be a challenging effort and it depends on the complexity of the information system. Tool supports are limited to identify the services in a program language independent way which can come in handy to automate or semi-automate the migration process. The granularity of the service design depends on the usage scenario. Not many best practises for designing services in the right granularity are available [28]. If it's too finely grained, then it will have performance impact. While if it is too coarsely grained, then not enough reuse of functionality is possible.

One has to anticipate potential consumers and usage patterns. Also, the consumers have to know the mechanism to compose services as well as to search for the services. The service consumers also need to be able to compose services when single service functionality is not sufficient to fulfil a request.

Essentially, the steps involved in the sub-process are as follows:

1. Understanding the “as-is” architecture
2. Identify the services within the architecture and annotate them
3. Document and list those services in a directory

We will use this directory to define the to-be architecture. Once we have a service-level understanding of the problem domain, we can identify the services that are good candidates to be run in the cloud. Then, we perform migration of the services that can be migrated based on KPI. The services provide software virtualization and this characteristic is leveraged upon no matter where they exist [23]. A single application can be composed of dozens of services hosted in a dozen different locations, on-premise and in the clouds.

Since all these expose information and behaviours as services, which typically use a common and standard interface such as Web services, they are location independent (cloud or on-premise), platform independent, programming language independent and User interface independent, if they are properly designed [23].

For all this to be achieved, we need to define our architecture as a set of services and classify these services whether they can be run in cloud or not.

This is done in the “Model Service” sub-process which is defined below [19]:

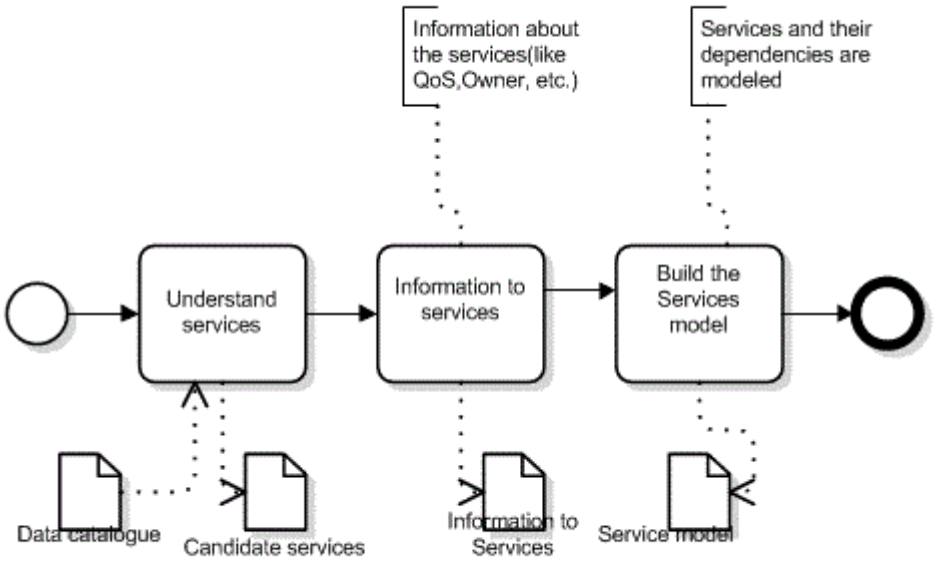


Figure 5-9 Service Sub-process [19]

We defined the data catalog in data sub-process. Services typically process information or are bound to data. Hence we defined it first.

Once we have understood the problem domain from a data-level with "Model Data" sub-process, we are mapping the problem in the service domain. At the end of this sub-process, we will have the core service model, which will then be used to make architectural decisions.

Essentially, we are listing all the services and meta-data about these services in the problem domain in a directory.

We perform this by looking at the data catalog that we created in the previous step. We identify the services that are bound to this data and list them as a service in a service directory. The services are decomposed (if needed) and ordered so the dependency among the services is properly understood. Essentially a service hierarchy is defined.

This service hierarchy is very important in a cloud scenario. One of the important aspects of this service hierarchy is the degree of coupling among the services.

Impact of Coupling on Cloud migration:

A strong coupling is a bottleneck in the architecture in a cloud computing environment, where the services are better leveraged placing them on different cloud platforms or more often in different datacentres.

Tightly coupled systems are dependent on each other [22]. Thus, changes to one component might need changes to other components. Loosely coupled architecture leverage on independent components and hence can operate independently even if some of the components fail. In a cloud scenario, failures might occur and the system can not come down in lieu to failure from a component. Hence, loosely coupled architecture is vital while migrating to cloud [19].

With the advent of technologies like Web Services, Enterprise Service bus and in general SOA middleware, loosely coupled systems can be realized with support for late/dynamic binding of components that have different security requirements, data format, protocols, etc. [23].

Compare this to tightly coupled architecture that requires you to bind the components at compile time or run-time respectively and requires that changes be done at all the components at the same time because of the dependencies. This is very difficult to enforce in a cloud scenario where the components are distributed across different platforms [19].

Having emphasized on loose coupling, there might be scenarios where tight coupling is needed. For e.g. a transaction system that requires a transaction to be completed within a certain time duration might require the components to be tightly bound. In those scenarios, those components can be tightly bound. [19]

However, tight coupling as an architectural style in a cloud environment should be more of an exception rather than the rule. The rule is, the higher the degree of loose coupling in the architecture, the more suitable the components are to be migrated to a cloud, if need be.

The degree of loose coupling can be identified by the following aspects:

1. Location independence: No matter where a service might exist, other components that need to leverage on the service can discover it and bound to it through the late binding process. This is extremely desirable in a cloud scenario where a service might need to be moved across from in-house to different cloud platforms. Dynamic discovery is the key component of a loosely coupled system. It should be possible to discover a service dynamically [19].

2. Communication independence: This implies that all components need not work on the same protocols in order to communicate with each other. This can be enabled by leveraging on Web service standards and other middleware technologies [19].

The need for loose coupled architecture in a cloud computing environment can not be over-emphasized. In order to leverage Cloud Computing correctly, loosely coupled architecture is a fundamental requirement except in some very rare scenarios. These details need to be captured in the service catalog to identify possible candidates for cloud migration. This catalog is built next which is defined below.

Service Directory

To build the service directory we need to externalize the services, understand the services. For that, similar to how we built the meta-data about the data earlier, we need to build a service repository with information about the service. There is no standard way to define service information. While WSDL (Web service Descriptive Language) provides information about the service, it does not include all the information that is needed to leverage on a cloud scenario [19]. For example, one might need information like rules to access the service, owner of the service, semantics, etc. These need to be defined in the service directory. The service directory is similar to a database/repository which contains information about all the services. This directory is the central node for the entire SOA process. This directory should keep track of all the information about the services, both on-premise and on cloud. This directory might even access the cloud systems and on-premise systems in order to discover necessary information. In due course of time, this directory will become the metadata of the entire enterprise keeping track of all the services and systems (both cloud and on-premise) in the problem domain.

The following attributes about the service need to be included in the service

1. Semantics
2. Purpose
3. Authentication
4. Dependencies
5. Owner
6. Performance attributes
7. Services leveraged within this service
8. Service levels [19].

This service directory will be used to identify and understand services in the problem domain.

Here, the transactions, the APIs, the Web services are externalized from the existing systems. The purpose of this is to list them, understand them, document them and link them to the data catalog designed in the data sub-process.

Implementation:

Aboulsamh [76] proposes that the service model can be implemented as per the assembly model of Service Component Architecture (SCA) Metamodel. The SCA assembly meta-model consists of a series of artifacts which define the configuration of a SCA system in terms of service components. SCA also enables legacy asset reuse. It does so by allowing SCA composite applications to contain both new services created specifically for the application as well as business functions from existing legacy assets. They also propose an object analysis model which is an extension of the UML use-model from which one can generate the SCA services model.

Aboulsamh [76] proposes using KDM to identify the business components in the legacy system and their relation with respect to our use-case.

5.2.3. Model Processes

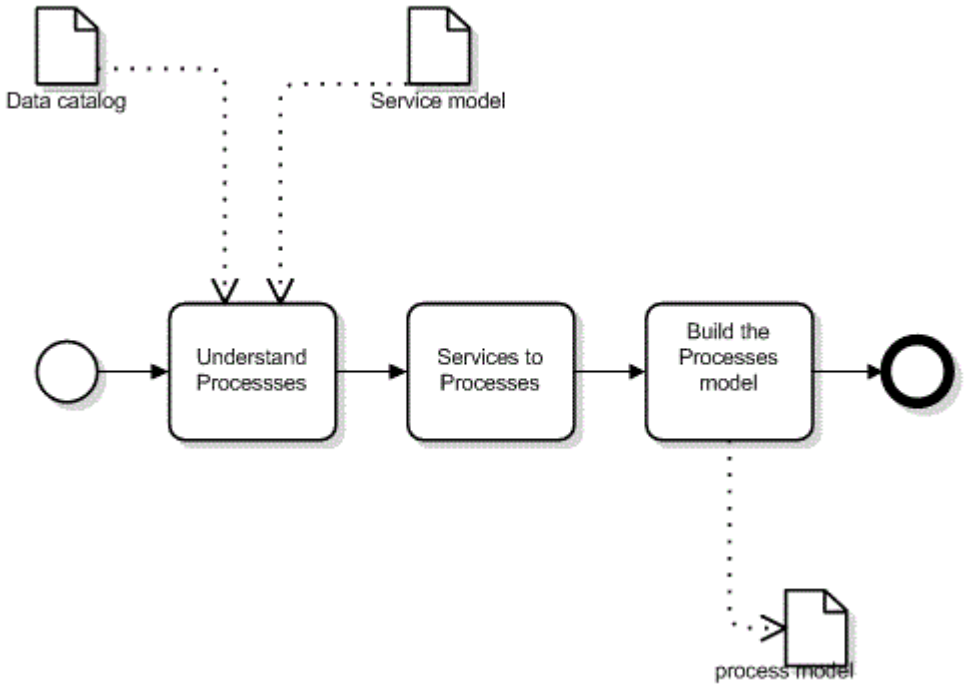


Figure 5-10 Process Sub-process

A business process can be viewed as the control logic on top of the services and systems that binds services into a multi-step business process that can carry out unique functions of the business process. This control logic manages the order the service is invoked, state of the process while handling any exceptions that might arise [19].

Now the business process management needs the following pieces to control the services involved and to provide the support for the necessary business processes.

1. Graphical Modelling tool: To model the business process.

2. Business process engine: To execute the business process while interacting with middleware that in-turn interacts with different systems. It also maintains the state of the business process.

3. Business process monitoring interface: It provides the interface to control and monitor the business process.

4. Business process engine interface: It allows the process to access the business process engine.

5. Application integration middleware: the components that help to interact and control the services involved [19].

Migrating processes to clouds:

The reason behind this sub-process is that we are trying to define the processes in the problem domain. Once listed, we can identify the processes that can be migrated to the cloud and ones which should be run on-premise. It should be noted that no matter where the process runs, the corresponding data, service can be run anywhere (different cloud platforms, in-house enterprise).

The process can be viewed as a service and its behaviour can be leveraged by other services. The only difference is that it's a lot easier to change a process with respect to changing a business rather than changing a service. Because changing a process involves changing the configuration whereas changing the service involves changing the code. Hence, if the services are designed appropriately, an enterprise can change its business process in an agile manner in tune to the changing business environment [24].

Building processes repository:

Like in services, data layers, the objective is to list the process details in a directory so that we can decide whether a process can be migrated to the cloud or not [19].

It involves the following tasks:

1. Understand the processes: Here, the processes (both the workflows as well as the abstract process models) are defined at a higher level.

2. Bind with the services: We bind the processes with the services to solve a business problem.

Why BPM and processes?

Processes are the central idea of SOA: the ability to orchestrate a business process using services from a configuration layer in response to change in business. Cloud platforms are one of the platforms on which services may reside. This means for a change in the business, there is no need to change the service. A change in the configuration layer of the process might be enough to align IT to address the change in business. This brings tremendous agility to an enterprise. This is essentially one of the most important benefits of service oriented architecture. This agility is more important than reuse [19].

Also BPM is used to orchestrate inter-enterprise processes, where the services involved might be running on different systems/platforms while controlling the information flow among the different services involved. The takeaway is that BPM can work with services deployed on different platforms. BPM leads to a process model which performs the orchestration needed to implement the necessary business function.

The use of a common process model that spans multiple systems provides many advantages:

1. Modelling: a common process across different systems (on-premise or cloud) automating the integration of the systems to react to varying business demands.

2. Monitoring: Analyse all aspects of the business by analysing the process in execution.

3. Business re-engineering: Based on the monitoring data, one can re-engineer the business process

Thus the process model can enable agility by moving the services involved across different platforms in an agile manner.

While building this process repository, we identify all the processes and model them if they are not already modelled.

Also we classify the process models as either internal, shared or external processes.

Internal processes are processes that should not be visible to external world or to business partners. Processes which are sensitive are marked as internal processes. These need to be reviewed carefully if they need to be deployed on a non-private environment.

Shared processes: These are processes that are shared among enterprises to automate business functionality. Most often, these are good candidates that can be migrated to cloud.

The migration to cloud computing will further underline the significance of BPM as we are trying to manage heterogeneous systems that may exist anywhere in the world, binding them together as a set of processes to perform a business function. A single process might be supported by a number of systems (on premise or cloud). Thus, a process will be the single architectural component that binds them together. More than deploying the process model on cloud, the main reason of developing a process repository is to develop the process models for all the business functionalities in the problem domain. Once a process model is accurately identified, we can span a process model across different platforms (cloud, on-premise).

5.2.4. Model Governance

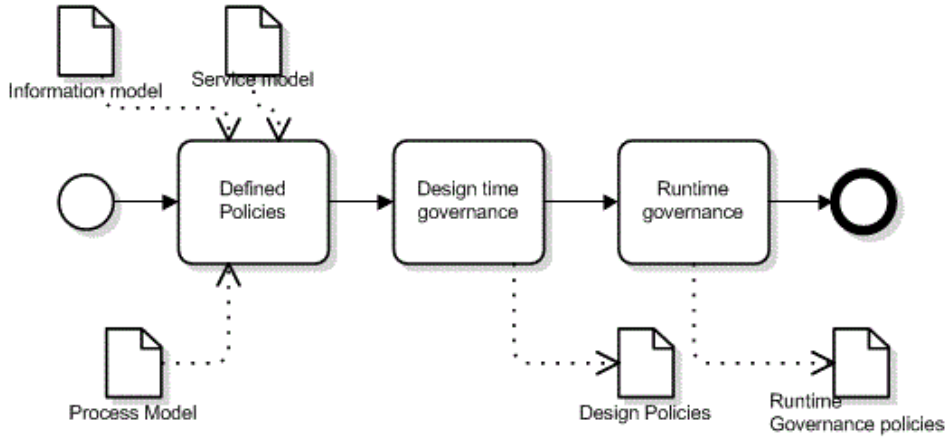


Figure 5-11 Governance sub-process

Governance is often the reason Service oriented architecture based IT projects succeed or fail [38].

Linthicum [19] describes governance as the ability to control changes to services and the usage of services. Cloud computing requires governance to be successful. An enterprise might contain hundreds or perhaps thousands of services and data elements. The way in which they are accessed, modified, added or deleted needs to be controlled. Hence, proper governance is of paramount importance. In SOA, governance of services is entrusted through the use of policies. In a cloud scenario, service governance is crucial as the service is deployed across different platforms (cloud or within enterprise) and the service is expected to provide the same quality of service without regards to the environment it's running on. Governance is implemented by means of policies.

Policies are declarative rules that define what can be done to a service by whom. They can be implemented during design time as well as during run-time of a service.

Design-time service governance usually leverages an integrated registry/repository that attempts to manage a service from its design to its deployment and typically not during the execution.

Runtime service governance is the process of enforcing these policies at service run-time. Here, the policy is run on a run-time governance platform that governs the service in execution.

Controlling the service on-premise is a challenge. So governing services which are deployed on cloud on which one has no control is even more challenging. However, this new state of

enterprise will be tremendously agile and hence mastering governance in a cloud scenario is one of the key aspects for a successful cloud migration. Here, the existing runtime and design time governance are identified and the policy artifacts are documented. They will be adapted for the cloud platforms in the to-be model.

5.2.5. Model Business Processes

A migration to cloud impacts an organization in a multitude of ways. For this, the current business processes need to be modeled to identify the roles of all the actors involved in the business process. Then each of these business processes is examined with the view of ROI for the enterprise.

For e.g. Mohagheghi et al. [61] discusses the business model of a CRM/ERP software vendor DISYS. Their software-support business process requires a manual installation of the software at each of the client's locations. While modelling this business process, the actors involved in the business processes are identified. The actors involved are the sales and the support team and the client.

The cost incurred by the client, the revenue for the enterprise and expense of the enterprise to maintain the sales team are modelled here. While modelling this business process, the need to keep track of the different versions of the software is analysed and identified here. All these cost and revenues will be used to make the case on whether to migrate the software to provide it as a SaaS solution will be made.

Similarly other habits in the enterprise are externalized here. For e.g. Mohagheghi et al. [62] discusses that the business process that DISYS undertakes to scale to new customers. The existing process is to buy new servers and install once for each server. Modelling this business process helps an enterprise to identify issues like low resource utilization, the capital and operating expenditure for each new customer. The cost-structure is again built and will be used to justify the reasons to migrate to the cloud.

Rashmi et al. [49] emphasizes that the increased network and the support requirement while an enterprise makes the migration needs to be handled. To plan for these sorts of issues, the current as-is state of the process of running the IT needs to be modeled.The IT architect/developer/manager is made responsible for estimating the cost of the in-house infrastructure, the maintenance costs involved etc.

All these business process models involved will be used to perform ROI analysis. They will also be used as the basis for modeling the corresponding models on the target environment, should the enterprise decides to make the migration.

Essentially, the semantics of the current business state as well as the IT state is modeled here. This will be used as decision support to make the case for or against migrating to a cloud environment.

5.2.6. Model Rules

The enterprise rules need to be modeled at this activity. There are many rules that might already be externalized as services in a SOA. So it might be logical to question the need for this activity. However Rashmi et al. [49] have pointed out that there might be rules which might not have been captured that are relevant for cloud computing. They say that for a country like India, cloud data centers might not be located within the country. While a rules engine might have rules to handle workloads, these rules might not exist in the rules engine already. This is crucial especially for companies in health care, finance as well as for government sector companies. An expansive list of such regulations including the EEA regulation and their impact has been described in [50]. The required level of security might not be modeled or might need to be updated.

Enterprises need to make sure they remain complaint to regulatory and governmental regulations as well as their local policies. Kaisler et al. [59] proposes that the organizations are unaware of the regulations as well as the degree to which they compile to these regulations. Before migrating to cloud, the stakeholders have to be identified and appropriate roles need to be assigned.

The constraints/issues of the current business and IT are externalized here. From these rules, code might be generated (semi)automatically. For e.g. OMG has defined SBVR as the standard to represent business rules. These rules might be maintained by the business user while the IT tools might help in validating and managing these business rules [29].

5.2.7. Model issues

The current enterprise vision is identified at the end of this knowledge discovery phase. The value of the existing systems to a business is determined. This is essential to perform a return on Investment (ROI) analysis. The problems in the business-IT alignment will be evident at the end of this task. Most often, the effort and the price of any software might not provide the business value it was expected to produce. For example, an enterprise might not even use all the features of a complex ERP solution that is just too costly to maintain as the business might have changed after its implementation. After identifying all these patterns and tendencies, the current enterprise vision will provide a Launchpad for a cloud computing journey. A set of Key performance Indicators (KPI) are defined at this state, which will represent the goals of the migration. For e.g. with the support of the knowledge from the different models below, one might identify a low resource utilization as the KPI of the current state of the enterprise. This will be used in the ROI analysis to decide whether a migration effort is needed. The KPI can fall into one of the following 4 categories

1. Time: e.g. The Time it takes to serve a request, the time to launch new markets(scaling time)
2. Cost: e.g. Cost of IT infrastructure, cost of salary
3. Quality: e.g. the customer satisfaction (%)

4. Flexibility: e.g. agility. This is often ignored.

This current enterprise vision will also be evaluated on different cycles to identify whether the cloud migration is performing well with respect to the Key performance indicators (KPI) defined in this sub-process. Similar to outsourcing model, a cloud migration might turn out to be a bad strategy for an enterprise, if not tracked properly [25]. For e.g. a cloud migration might turn out to save costs for a problem domain initially. However, the cloud platform might go down and the service availability might be compromised [9]. The initial migration quickly becomes a bad step if the disruption causes more damage to the enterprise or it might force the enterprise to make use of features which were not considered while doing the cloud migration initially. We have seen the cloud providers increase the price of the cloud usage for the customers which might also put the initial cloud migration decision as a bad strategy [13].

So this current enterprise vision is a crucial element for an enterprise's cloud strategy. Yet, monitoring and controlling business process is a challenge. Automating the monitoring of the Key performance Indicator might be crucial so an enterprise can actually focus on the business goals and the corresponding IT architecture rather than focusing on getting the accurate data from different data sources to track the KPI. An appropriate monitoring solution might need to be employed to perform course correction in an agile manner. Kintz et al. [18] proposed a semantic dashboard description language to track the Key performance indicators that helps in tracking the goals without focussing on technical aspects of controlling and monitoring infrastructure.

5.3. To-Be model

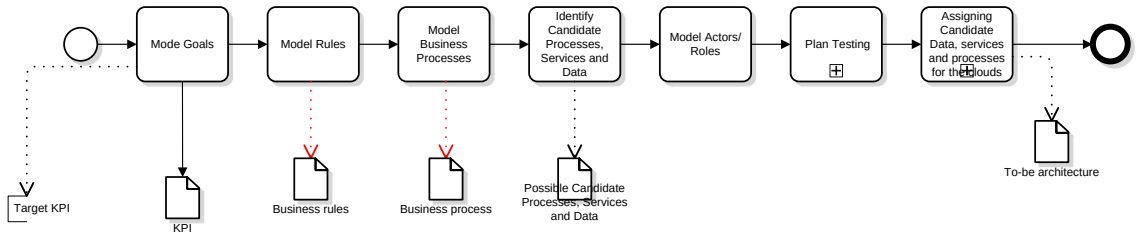


Figure 5-12 To-Be Model

5.3.1. Model Goals

The Goals Model is used for describing the strategic goals of the enterprise along with the issues associated with achieving these goals. It's a collaborative process with different views and that makes it difficult to organize and understand.

In this sub-process, we look at the "issues model" to define the target KPI for the enterprise. In the target KPI definition, there might be big strategic goals which might take several years to achieve. Simultaneously, we define KPIs which can be attainable in the immediate future. We emphasize on a phased approach to migration. The phased approach enables lower risks, higher returns and faster delivery [45]. A set of systems might need to be selected for cloud migration in a particular timeframe with a defined budget and return on investment expectations.

In the key-performance Indicator, the value of agility and the ability to scale needs to be externally captured.

This is an extremely important step in determining the Return on Investment on a cloud computing journey. The costs involved in actually porting the application, or the cost involved in modifying the application to make use of cloud resources, security issues, compliance issues and the cost of trusting another organization, where an enterprise might not have any governance need to be considered in performing the cost-benefit analysis while defining the target KPI. There needs to be techniques to establish and document the business case to quantify the values of agility and scalability for an enterprise [19]. Also, there should be a funding model for the provider and consumer of shared services. This is complex as cost and benefit of services are attributed to different business units [44]. This is one of the major reasons for the resistance to migrate to a more efficient environment.

Most of the organizations that look at cloud computing look at the single dimension of cost. The reality is much more complex.

One needs to understand the new **security** issues that cloud computing introduces. The loss of **control** of data, services and processes while running on cloud might cause service issues. The cloud provider might close a client's account as one might accidentally violate some policy or it might happen that the cloud service provider goes out of business [8], leaving the cloud consumer to scramble for another cloud provider. These are risks that need to be factored in while finding an appropriate cloud provider.

It might be the case that some services are no longer **cost**-effective while running on a cloud provider [13]. It might be the case that running it locally will be cost-effective in certain scenarios[10].

The lack of **open standards** might need to be taken into account if a system needs to be migrated back in-house or if the system needs to be migrated to another cloud provider.

There might be **compliance** standards that organizations must adhere to and this needs to be taken into account while choosing a cloud solution.

Also, **Service level agreements** need to be considered while determining the KPI. Though, many cloud providers do not provide them or there are misinterpretations in this area, this need to be externalized as well in the KPI.

Essentially, an enterprise needs to externalize all these issues and attach a priority/value for each of these issues. The following dimensions need to be considered while determining the KPI

1. On-going operational cost reduction
2. Value of Capital expenditure preservation
3. Value of scaling(up and down)
4. Value of shifting the risk
5. value of agility
6. value of reuse

For example, one needs to carefully analyse the value of existing infrastructure (hardware and software) that has been already paid for [45].

The risk of leveraging a platform that the enterprise does not own comes with a cost. The benefit of a faster time-to market needs to be weighed in. So there are several hidden benefits and issues that need to be identified before fixing on the KPIs for the organization. There might be costs involved in building gateways to integrate with in-house components. Also, there might be costs involved to modify components in order to make sure that the compliance requirements are met, just as like in an in-house environment.

In addition to the goals, problems and opportunities related to the goals are externalized as well in the goals model. As we perform goal modeling, we identify some key performance indicators for this new "to-be" state. These indicators help the enterprise to track its progress over different cycles of cloud migration.

Some example goals and the corresponding KPIs are given below:

Serial No.	Goal	Priority	Problems	KPI	Opportunity
1	Reduce power and cabling bills	High	Servers are decentralized	Server Utilization (%)	Server virtualization technology
2	Website is available for customers 24*7 without disruption	High	Not enough capital to buy redundant servers	Service Downtime(minutes /hours)	Cloud Computing technology
3	Serve 2X	medium	Datacenter	Number of Requests	Cloud

	customers that will arrive due to mobile strategy		costs are too high	to the service	Computing technology

Table 1 Goals Model

Some of the driving questions that can be used to perform goals modeling are

1. What are the strategies of the enterprise in this problem domain?
2. What do we want to achieve?
3. How important is it to achieve this goal?
4. What are the problems that hinder the enterprise from achieving these goals?
5. What are the opportunities that are available to reach there?
6. What are the Key performance indicators that will help us to track our journey towards the goal?

Erl [32] suggests that the return on investment (ROI) over the long term needs to be estimated as that will provide the basis for allocating budget for migrating to a SOA environment. Over a short term, the costs might run higher than the value of the restructuring, but over a long term, the value will far exceed the initial cost involved.

Implementation

For e.g. Klems et al. [59] provides a framework that helps decision makers to compare the cloud computing costs with traditional in-house computing. SMART project provides a framework for comparing the value of legacy solutions to migrate to a SOA environment [42]. This is where application portfolio management is performed to evaluate the value of the applications

5.3.2. Model Rules

Now that the goals are modeled, in each enterprise there are some rules that the IT needs to adhere to support the business. Bubenko jr et al. [15] describes that it can be viewed as "precise statements that describe the way that the business has chosen to achieve its goals and to implement its policies or, the various externally imposed rules on the business, such as regulations and laws." They might trigger business processes in the business process model which will be defined next.

Business rules also may require certain functionality from the information system.

For e.g. a video streaming service provider might need to provide streaming to different customers differently. For premium customers or customers who want to stream in the highest clarity possible, business might have rules to deliver the service using certain rules(for e.g. using a content delivery network close to the customer to serve these customers). Now, this might be changing every once in a while as the business environment changes. Now, if this is externalized, Business consumers need not go to IT every time the business changes. For e.g. it may so happen that the business is very competitive and that the enterprise decides to use CDN for long-standing normal customers as well. Now, for this change, if the Business rules are implemented using a business rules engine, then this change can be implemented at real time.

Another example with cloud might require storage and computing takes place in a distributed manner. One might want to make replication across different zones (similar to Amazons Multiple region set-up) to ensure high availability. It might happen that some resource (data/process/service) is not allowed to be stored outside a state/country due to regulatory compliance. Now, this again might change in due course of time and hence can be adapted by using this rules model.

Some of the driving questions that might be useful when modeling business rules might be

- Are there stated rules and policies within the company that may influence this model?
- What are the rules necessary to achieve the business goals?
- How can the enterprise conform to the specification of the rule?
- How do you validate that a rule is enforced? [15]

5.3.3. Model Business Processes

Bubenko jr et al [15] proposes the following way to define the business process model.

Here, the new business process of the enterprise is modeled. It makes use of the business architecture, the rules model and the business goals model. It's a perfect place to design business process Re-engineering.

The existing "as-is" business process model is analyzed statically or dynamically to optimize the business processes. The optimization might involve elimination of unnecessary activities, automation of existing activities, re-sequencing activities and parallelism of a set of activities. It will be re-engineering based on the strategic goals defined in the "goals model". These business processes are designed at a high-level view. It only lists the major activities, organized units involved etc. These activities will be refined later in detail. A Process Map (or Chevron Diagram) might be used to provide a high-level and coarse-grained view of a

large organization. This diagram might represent groups of processes of different units of an enterprise. This results in input for concrete process modeling [24].

For e.g. a Video Renting retailer might have designed the goals and the Rules model to change his business model from renting DVD to streaming online. It's in this model, that the business is formally re-engineered. The new business process is designed identifying the actors and the interaction among the actors, etc. Each of the processes is modeled by the goals defined in the enterprise goals model. These processes are designed at a higher level in the business domain. They might be translated into a business process when it will be implemented in the Information technology domain.

Some of the driving questions that might help in designing this model are:

- Which are the main processes of the enterprises?
- How are these processes related?
- Why is this process needed?
- Which information and material flows does is it need?

5.3.4. Identify candidate processes, services and data

An enterprise might have lots and lots of applications. Identifying the right candidates to show value in cloud computing migration might be challenging. The business architecture of the enterprise as well the business process model is used to perform portfolio analysis to identify the right candidates. The basic idea here is to identify candidates that are of strategic importance for the company.

Business Capability based analysis of the portfolio enables an enterprise to build application taxonomies which helps to identify the candidate services that are strategic for the enterprise. For example, it is not uncommon to identify that an organization's investment strategies, while sound at a technical level, do not map well to overall business strategies. For that reason, application portfolio analysis should be a continuous effort on the part of both business and IT [17]. Application portfolio analysis should be part of both the strategic planning and the project lifecycle of an organization. The IT organization should be evaluating the technological reasons that are driving application portfolio change, and the business should be identifying ideas about where they would like to invest and how IT needs to be aligned in the new initiative. Business capability-based portfolio analysis helps enterprises to keep its software portfolio and business strategy aligned. For e.g. a bank's projects -- whether infrastructure- or process-oriented -- are mapped to business capabilities and those are ranked based on their contribution to strategy and business value. Once this is done, then decision makers can use that as a starting point for return-on-investment discussions. Here, in this task, we try to reinforce the alignment between IT and business.

Here, the business architecture is used to identify the right candidate processes, services and data.

After identifying the current state of the enterprise in the "as-is" model and modeling the new enterprise goals model, business and IT work together to identify the resources that might be probable candidates to be migrated to cloud to realize the new enterprise vision.

Now, this is performed from process to the services and the data level. This is done in this way because after identifying the processes, services and data, taking into account the new goals model, some processes can be ignored straight away as they are not strategic for cloud migration. Thus, we do not have to analyse the data and the service associated with the process.

For e.g. a video streaming service provider decides its new business goal is to stream videos over mobile and internet rather than delivering DVDs by mail. It might not want to perform migration of some services which are related to physical DVD delivery as it will not be of strategic importance to the new business goals. Hence the processes might well be ignored and since the process is ignored, the corresponding service and the data need not be analysed as well.

Also, it might be the case that due to business rules or company's strategy, the resources are marked whether it can be migrated to a public cloud or a community cloud or need to be inside the enterprise. At the end of this task, some processes, services and data are ignored from the problem while some other resources are marked for appropriate levels of cloud type that it might be possible to migrate to.

S. No	Resource	Possible cloud candidate
1	DVD availability checking process	No
2	Online Streaming processes for customers	Yes(Public, Private, Community)
3	Content Locator Service, Content Streaming service	Yes(Public, Private, Community)
4	Related video Content	Yes(Public, Private, Community)
5	Customer credit card details	No

Table 2 Identifying candidates for Cloud

5.3.5. Model Actors/Roles

Smith [28] proposes that when a change needs to be made to services, it needs to be communicated and coordinated with potential innumerable number of users without causing service disruption. A change in service might lead to changes in a lot of dependent services. Hence one might need to provide multiple versions of the service to avoid service disruption while at the same time remove the redundant service within a suitable time-frame. For this, it's important to assign appropriate roles for the actors involved in the services to take appropriate action during the lifecycle of a service.

Schepers et al. [31] proposes that ensuring quality of service is hard to implement. Assigning roles is important to take corrective actions in scenarios when the qualities of services are not met. Compliance to standards needs to be enforced by the appropriate authority.

All these issues are not strictly IT related and they need to be solved collaboratively working with the business. Smith et al. [28] proposes that the problem that needs to be solved is SOA governance and not only IT governance. Schepers et al. [31] proposes a 7 phase SOA governance lifecycle model that addresses the business specific demands of this architecture. It can be used/ extended to implement governance in the cloud.

A team that collects information across different migration projects and drives incorporation of these experiences is generally formed to drive the project. An example of such a team is the so-called Center of Excellence. The CoE might train both service producers as well as consumers on the best practices that need to be followed on an enterprise. They can notice problems easily and identify these patterns and ensure that solutions to similar problems are incorporated in successive migration cycles. This team might be held responsible for enterprise wide governance policies whereas the individual business units might be responsible to implement on a local level with respect to its service and process requirements.

Similarly, the data catalog needs to be maintained up-to-date while ensuring no disruption to services. The identity of the consumers of the service/data/process needs to be controlled appropriately by the respective owners while ensuring quick access using role-based identity management. And this needs to be automated to scale to enterprise levels [19].

Schepers et al. [31] proposed that grouping of services which was performed in the "as-is" state might be helpful in reducing the number of artifacts to be considered for service governance. Due to the Distributed nature of service development process, Or due to insufficient knowledge of the services, there might be duplication of services or rogue process (unregistered process that can not be governed) might get created and that again leads to maintenance issues while affecting the agility of the enterprise.

WS-Policy framework provides a XML-based language to describe the policy while WS-Policy attachment explains the mechanisms to attach the policies. Various WS-* defines domain specific Policy Assertion Language such as WS-Security Policy, WS-RM Policy, WS-Transactions and WS-Addressing Metadata. These policies are enforced at runtime by a policy enforcement point. They might be deployed as part of the service or at a gateway or can be implemented at the message transport layer. The WS-policy does not emphasize on the position of this PEP. Hence, one might find an appropriate SOA infrastructure product that best fits its needs. While Services promote agility, runtime governance of policies provides control and is crucial for SOA adoption [31].

Define policies

These are policies that are defined to control the resources (data, services, processes). Some of them might be electronically enforced by governance technology, while some of them are not.

There are some policies which are common to all the services and some which are specific to particular services.

For example,

A generic policy could be that all changes to the processes must be approved by a business leader.

More specific policies might be designed for more-grained services. For example, a core service which will be used by many processes might have a stringent requirement that it returns within 0.XX seconds. In this way, services can be designed.

Design policies

We design the policy for the services describing who is authorized to access the service, how the access is logged and how other services/processes/data are bound to this service, etc. These policies are designed to be executed in a runtime service governance technology platform.

Implement Policies

Policies like services need to be designed, developed, tested and deployed meticulously. In this task, we are implementing the policies which include testing and deployment

Runtime governance technology provides features to test policies as they execute along with the services on a test platform after sufficient testing, the policy can be implemented on the production environment. Smith [28] proposes there needs to be a validation model to ensure the runtime governance is in compliance with the service policies. These might be needed to audit the behaviour of services that are called by a user request for compliance or legal issues.

Defining and maintaining the policies is a challenge as services evolve with respect to business. Yet, this needs to be maintained diligently in a cloud scenario where the services might need to be moved across platforms frequently. A good runtime governance solution is crucial to control the execution of services, as it reduces risk and saves avoidable costs. The run-time governance needs to ensure governance as the services evolve in a service oriented organization.

Smith [28] proposes that identity management in multi-organizational SOA environments is a challenge that needs to be addressed in governance. Schepers et al. [31] identifies ensuring quality of services at run-time is hard. In order to address these challenges, SOA governance needs to involve business and operations people instead of restricting it into an IT governance problem. It's because IT translates business activities into services. Hence these need to be modelled in association with the business and it will be performed in the "To-Be" model [28].

Also, because the services are dependent on other services, a single service failure might be critical to complete a business process. If a core service goes down, that might result in a substantial amount of revenue for the enterprise. Hence, the services need to be monitored continuously. However, Service monitoring might have an impact on the performance of the system. So this needs to be optimized so that the services are monitored frequently without affecting the performance of the service. However, in a cloud scenario, a cloud provider might not provide the service level monitoring that an enterprise might need. Some service monitoring might need just the status of the service while some services might need more granular monitoring like CPU utilization, Database load, etc. [19].

Hence, the granularity of governance needs to be considered while choosing an appropriate platform for a service. This again needs to be collaboratively defined in this model.

An enterprise might need certain actors to develop IT skills on cloud technologies. These early adopters might need to be identified to develop/deploy and test the cloud service. These actors are associated appropriate roles in this sub-process.

5.3.6. Plan Testing

This modeling is done to develop a test plan for the to-be system. Similar to test-driven development, this migration is also test-driven. Here, when testing is focused on before development, many of the issues are avoided before they are made and realized.

Migration to cloud will most likely involve differences in performance as well as functionality. Most often, this step is often ignored which might lead to a gap between understanding the business and the IT due to performance requirements. However, before IT jumps off on a cloud-computing journey, this step is needed to ensure business/IT alignment. Essentially, the business architecture needs to be referred to identify the performance requirements of the application portfolio in the problem domain. This is again crucial for a successful cloud migration.

For example, an online streaming service provider migrating to the cloud to scale to more customers might have focused on scalability and cost and might have ignored performance requirements. Krishnan et al. [16] shows "that an increase in the startup delay beyond 2 seconds causes viewers to abandon the video. Using regression, we show that an additional increase of the startup delay by 1 second increases the abandonment rate by 5.8%". This might be used as a technical and functional requirement for certain services.

Since Businesses and IT need to work together to arrive at these requirements, a test plan is formed in this activity in an agile software development manner. The test plan is made before selecting the service and migrating the applications to the cloud [19].

A mix of private and cloud computing systems and services makes testing a complex proposition. With some of the systems not under direct control of the enterprise, it is even more challenging.

We need to approach testing on multiple dimensions including the services, governance, processes, and data and so on. One has to make sure that performance and stability are tested in addition to regular functional testing on services which might be hosted on remote systems.

Since most of the services might not be under complete control, it's very important to design for all possible ways of failure and test them. Now, this aspect of testing was never considered in the "as-is" state without cloud services. Since the enterprise might not own the platform on which a service is deployed, it might not be possible to perform extended white box testing as if the service was deployed inside the firewall.

Traditionally, an enterprise never tests systems on which it has no control and that behavior needs to be changed. We also need to make sure that the cloud delivered service is as per the agreed terms. Testing needs to make sure that the services meets the agreed quality of services. This validation of the QoS of the services is another new aspect that cloud introduces to testing [19].

Now, Linthicum [19] proposes one way to approach testing is to break the architecture into different domains (i.e. services, governance, process etc.) and test each domain with the appropriate tools available.

Thus it involves

- Service-level testing
- Process-level testing
- Governance-level testing
- Data-level testing
- Integration-level testing
- Security-level testing

Service-level testing:

While performing service-level testing, the degree of loose coupling needs to be taken into consideration. The service must be tested to run both independently as well as part of composite application with the required quality of service.

Services must be tested to make sure that the services are neither too granular nor too monolithic. Too granular services will impose communication overhead whereas monolithic systems are not optimal on service oriented architecture. Hence services must be tested for reuse. This ensures that the service-size is appropriate so it can be used by other services.

Services must also be tested for heterogeneity as services by definition have to provide virtualization of software. Hence, clients from different platforms independent of their location need to access them.

Security-level testing:

With cloud, security level testing comes into the picture. Its best to approach this by listing the use cases, build a test plan around it and identifying the possible vulnerabilities.

Since a service that is deployed inside the enterprise might be accessible by a service deployed on the cloud, new types of information risks exist and need to be taken into account while building the test plan.

Integration-level testing:

It's important to test both the network and the protocols further up in the layers to ensure that the services can communicate over different networks on different platforms while using different transport protocols.

Following are some things to look for in integration-level testing:

- Can communications be established with late binding, meaning dynamically as needed?
- Is the integration stable under an increasing load?
- Is the transmitted information correct in semantics and content for the service or applications?
- Are the security mechanisms working properly?
- How does the SOA using cloud computing recover from application, database and network failures?

Data-level Testing:

Here, we test the persistence layer of the architecture, typically the databases for their performance and stability. The data latency and integrity issues need to be considered for the problem in-hand.

Similarly Process-level and governance level testing need to be considered and included in the test-plan.

Since the services might/might not be under direct control, black box test plan needs to be drawn in detail. To whatever detail possible, appropriate white box test plan needs to be developed.

The steps involved in this could be

1. Create the test plan and make sure to define the purpose of the service, who is the user and so on. We must define the input, output and the performance expectations.

2. Identify the data leveraged by the service and design for boundary cases and failure scenarios.

3. Identify service hierarchy and simulate service failures on services that leverage on other services.

4. Test the service across different interfaces (human, computer programs) and have them listed.

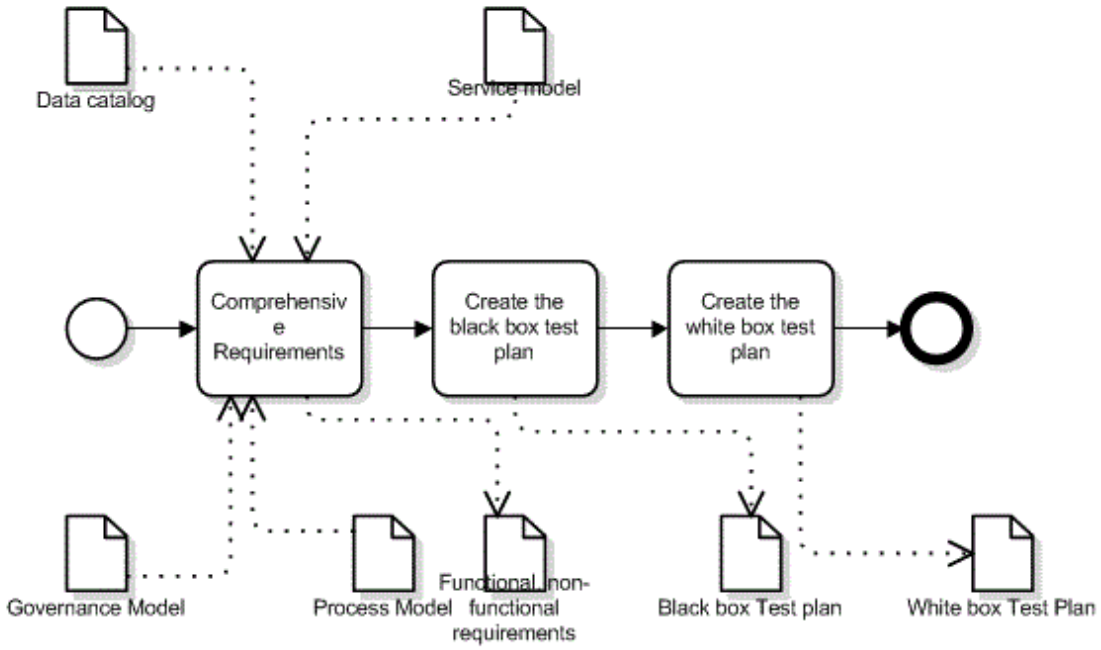


Figure 5-13 Plan Testing

However there are still some issues that need to be considered.

1. One needs to simulate the failure of one of the components and test the architecture of the entire system. This has not been the focus on on-premise systems. Tools like Chaos monkey need to be built to test the resilience of the architecture [30].

2. Automated testing of different services to compare against benchmarks needs to be built to compare and select appropriate services.

5.3.7. Assigning Candidate data, services, processes for the clouds

The Goals model has identified the business goals and has defined the KPI for the to-be state of the architecture. The business rule modeling has defined the constraints (regulatory, policy, strategic, security, privacy etc.) for the to-be architecture. Then, the business is modeled again which might involve business process re-engineering. Then the business and the IT work together to apply the rules and the KPI on the technical and business architecture to identify the possible candidates for cloud migration. The difference between this task and "Identify candidate processes, services and data" is that there in „Identify candidate processes, services and data" model, the semantics of the services, processes and data from the business architecture is used to identify candidates, whereas here, the candidates assignment here is strictly technical. The test plan has defined the non-functional and functional properties of the to-be system. The candidates selected above are evaluated based on the test-plan's functional and non-functional properties and appropriately the corresponding resources are marked as candidates for possible cloud migration. Essentially a high-level design of the to-be state is identified.

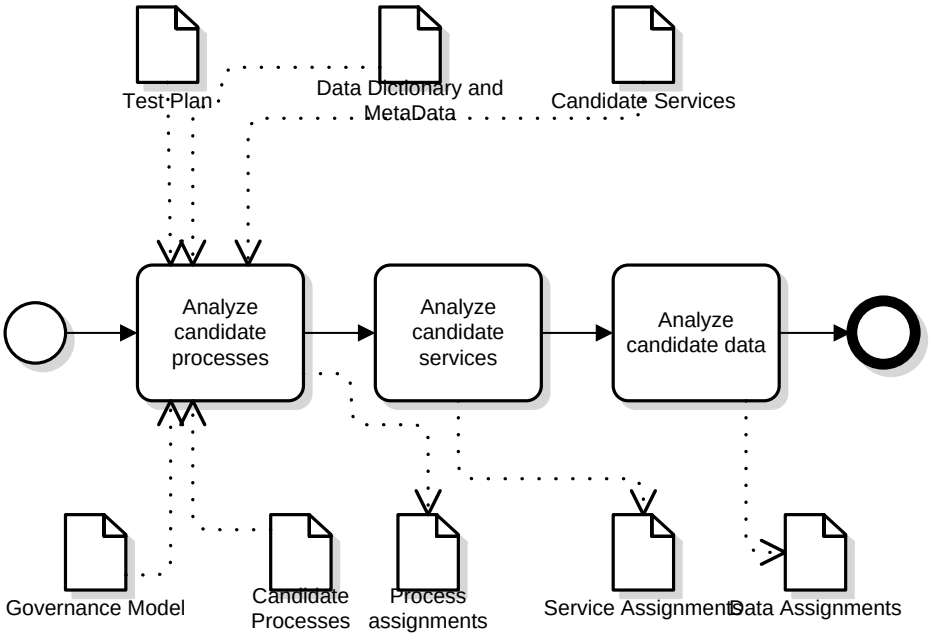


Figure 5-14 Assigning Candidate data, services, processes for the clouds

It might not be possible to have a detailed state of the to-be architecture as for the same component, 2 or more distinct routes might be possible. For example, an email server to be migrated to the cloud can be realized by leveraging on software as a service offering or it can be implemented leveraging on a private cloud infrastructure. It might depend on the user acceptance test after the service has been migrated to the cloud.

Similarly a data migration might require migration to a public cloud based NoSQL offering or might require employing the same RDBMS over a private cloud.

All the issues related to migration to cloud have been externalized and these detailed requirements are then handed over to the IT to define the "to-be" state of the architecture. In this task based on the models above, a corresponding "to-be" state of the architecture is defined by assigning the appropriate resources for cloud migration. While the assignment of a component to a cloud type (private, public and hybrid) or to a cloud service model (SaaS, PaaS and IaaS) depends on the component and the context of the enterprise, Linthicum [19] presents below a generic set of guidelines to assign the resources to the cloud migration.

As cloud computing matures, its normal to distribute resources across different cloud platforms. However, if the architecture is not properly designed, this might introduce new problems, like failure of a single component on a cloud platform might bring the entire application down. Hence, availability might be compromised by migrating to the cloud. The architecture might need to be designed to be resilient against these sorts of failures.

Also the performance needs to be considered while considering resources to be migrated to the cloud. If the services are too finely grained, there will be loss in performance. Now, with cloud, the services might be distributed outside the enterprise. This might not be acceptable for many applications and hence needs to be taken into consideration while considering applications that can be migrated to cloud.

It's a good idea to start with processes, services and data that does not require critical level of security. Since Cloud deployment might mean deployment on new platforms, it's optimal to perform migration of the resources which are not stringent on the level of security. If the risk is too high, the appropriate resources should be marked to be deployed on a private in-house platform.

It might be a good idea to assign applications which are relatively new for cloud migration. Legacy applications might require rework to migrate to the cloud. This redesigning increases the cost and risk of the migration processes. Most of the legacy systems are built as monolithic, tightly bound applications. However cloud is a distributed system and the applications need to be designed in a distributed way. This requires substantial effort and migrating to cloud with the same architecture does not take full advantage of the cloud platforms.

If the Data, Services and processes are largely independent, it is easier to migrate to the cloud. However, if they are tightly coupled, then it will be very difficult to do so. The reason is that with cloud, failures might arise which are not in the control of an enterprise as the service might reside outside the firewall. If the architecture is tightly coupled, then the entire system will come down if one of the pieces falls. With cloud, there are multiple pieces moving and the system must be independent to failure of one or a set of components.

Also, it would be a nice idea to pick components/services that do not have a stringent security requirement.

Also, it's a nice option, when web/internet is the platform. Any application that needs to tap into cloud resources/API might be a good candidate for cloud migration.

Also, when the applications are new, it might be a good idea to assign them to be deployed to cloud.

If an organization needs to have 100% control over certain components and can not be outsources to anyone who is less than 100% reliable, those components should be left alone.

If the applications leverage on native interface like system level calls, then moving those applications to cloud might not be a good option.

However, those were just the guidelines and there are no hard and fast rules. Essentially, an organization has to understand its internal architecture, identify its "to-be" state, and then use SOA principles as guidelines to migrate to the cloud, while aligning itself to the enterprise goals.

Implementation

Here, while defining the to-be architecture, the principle involved is to describe a service enablement option to transform a legacy model element (or group of elements) to a service element (or group of elements). For e.g. a batch import facility that directly accesses/updates a database could be migrated to a database adapter service for database access and a business service containing data integrity rules.

The service model in the "Model Service" process and the legacy model discovered using KDM is used to service enable the component under consideration.

5.4. Change process Model

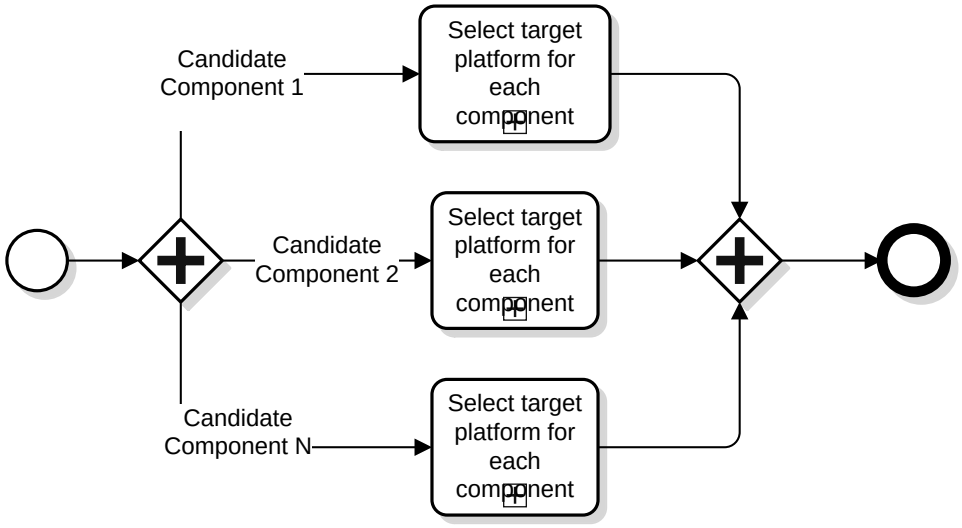


Figure 5-15 Change process Model

Here, we are implementing the architecture that was built in the "to-be" model. The technology and the cloud computing providers might change frequently but the architecture and the change process can be used.

Also, the selection of the cloud providers is done at the last stages of the migration project as the focus of the migration should be on the architecture and not on the cloud offerings and the related technology with it [19].

For each of the components that are assigned for cloud in the "to-be" architecture, an appropriate platform is chosen in parallel. The selection of a target platform is dependent on the intended goals of the enterprise for the process in concern. For e.g. an enterprise which is geared towards Green Computing might choose a platform which runs in the most optimal energy efficient manner for certain processes while for some other processes, the platform which runs in the most energy-efficient manner.

5.4.1. Select target platform for each component

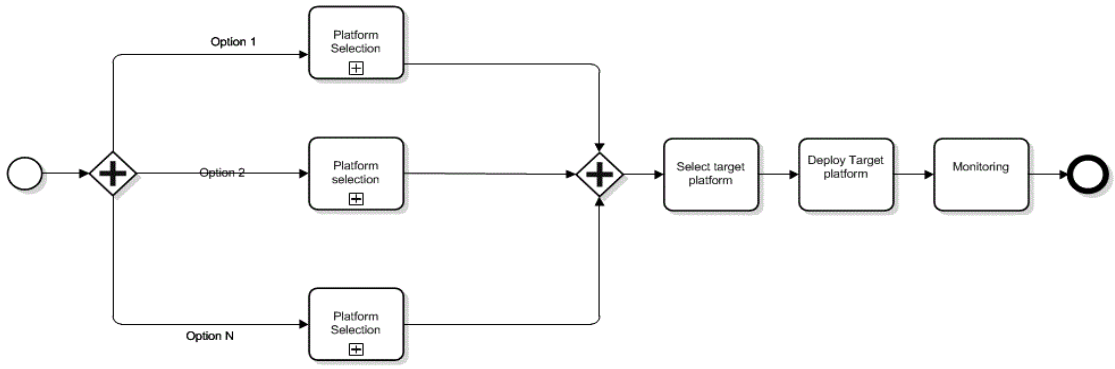


Figure 5-16 Select target platform for eac component

For each of the components, multiple cloud service types (SaaS, PaaS and IaaS) can be leveraged to realize the functionality of the component. For e.g. a database that needs to be migrated to the cloud might be leveraged by using an IaaS or by using PaaS. Similarly, the same data migration might be realized by migrating to different cloud types. So the platform selection depends on the to-be architecture and goals and other constraints.

For each platform (for e.g. an IaaS platform), there might be multiple vendors that need to be considered.

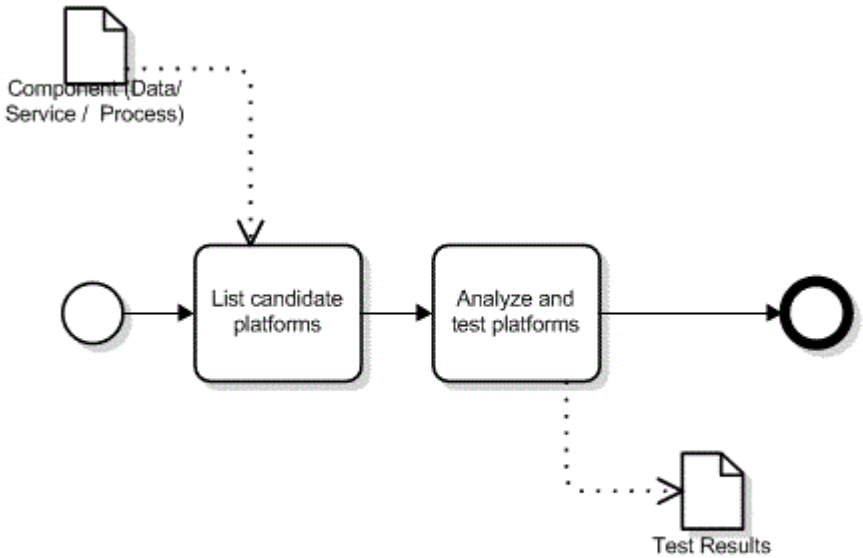


Figure 5-17 Select Target Platform

For this, first the candidate platforms are listed. For this, one needs to have a good understanding of the architecture as well as the available cloud offerings that fit well with the requirements. Different categories of the cloud platforms and different vendors corresponding to those platforms are listed.

The component in consideration is then tweaked/re-coded for the platform, tested on the candidate platforms and the results of the migration in terms of the KPI and other requirements are measured from this platform selection sub-process. These performance requirements come from the test-plan described in the to-be model.

Then the appropriate platform is chosen. While choosing the platform, the performance and the test results alone are not the only criterion. The agreement might need to address several Quality of Service rules and constraints that were identified in the to-be architecture need to be taken into consideration. [18] It might be the case that the cloud platform provider might be a perfect fit. But the cloud provider might one day go out of service. So the probability that a provider might not go out of business or the open standards of the provider might become a factor in selecting a platform.

Once the platform is selected, the components are successively deployed one after another and perform sufficient integration testing. Once sufficient amount of testing is done, the request to service might be served completely by the cloud deployed component. It is advisable to monitor the service closely by setting up automatic alerts as well as recovery mechanisms in case of failure of a component. This monitoring brings about new challenges as the monitoring platform might be outside the enterprise. Hence, here, the monitoring data might be provided by the cloud provider. IT will need as an auditor and audit the service to make sure that the services provided are as agreed upon earlier.

Now, the migration does not end with monitoring the service on the target cloud. The KPI of the enterprise is again measured to identify the results of the cloud migrations. If the expected results are not achieved, corresponding to-be architecture is again defined and the change-process model is called to migrate to the cloud. If the migration was initially successful, more services might be migrated to cloud to reach the long-term KPI of the enterprises.

In this next 3 units, we provide sample use cases for the proposed cloud migration process model.

Now, these cases are chosen to show that each of the three types of modernization that ADM (Architecture Driven Modernization) defines can be handled by our process model efficiently.

6. DISYS‘s Legacy to SaaS Migration

This use-case is an example for the Business Architecture Driven Modernization where the existing solution is an in-efficient legacy software solution while the target system is a SaaS solution built applying Service oriented Architecture methodologies.

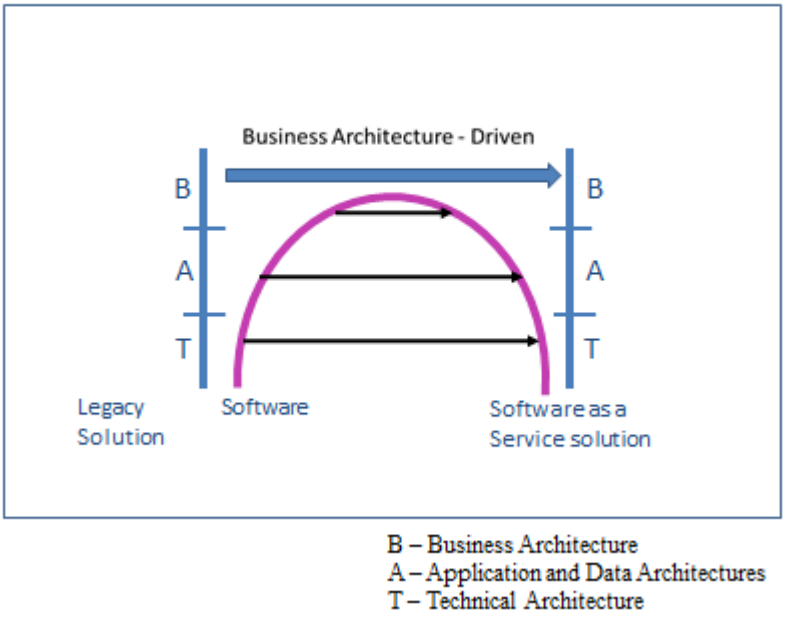


Figure 6-1 DISYS Cloud Migration

Mohagheghi et al. [61] discusses this case study on their paper about the “software engineering challenges for migration to the service cloud paradigm“.

DI Systemer (DISYS) is a Norwegian software vendor in the (Enterprise Resource Planning) ERP/CRM (Customer Relation Management) domain in the SME sector. Its products and services are CRM, accounting, payroll, invoicing, webportals and Application service provider.

Its application Portfolio has been built over decades with different tools and languages. The tools/languages used are COBOL, Delphi, C#, .Net, ASP, .NET and UML.

The products are used in different runtimes: (i) as stand-alone desktop application, (ii) as traditional client/server software solutions, (iii) in virtual machines executed in DISYS ASP or in their own datacenter. The resources are not shared i.e. one installation for one client. Its

software portfolio has a conglomerate of software components of different kinds with a high degree of interdependencies and few services. Most of the components have been built using legacy tools and technologies.

The software portfolio requires a modernization as there are issues with legacy code, no proper documentation and some employees have left the company with the knowledge of the code. Also, adding new functionality takes a lot of time. They have been following a bottom-up modernization approach for the last 10 years by rewriting COBOL code part by part using MDD technologies using UML to generate Delphi code. However, still a considerable part of legacy code needs to be rewritten.

Below we see how our process model can be applied to define the next steps in their migration approach.

Since we propose an agile phased migration approach, a part of the product portfolio needs to be chosen as a problem domain. In our case, we choose the accounting software information system as the domain we want to focus on.

6.1. As-is Model

The knowledge discovery process visually represents the COBOL legacy code. It will enable us to inspect the UML model to understand data structures behind the code as well as the semantics as well as from the employees.

6.1.1. Model Data

The data catalog for the reporting software is built. It has been identified that it needs the data generated by the accounting software. Since the reporting software and the accounting software are tightly coupled, the data is stored in the client's location. The data required by the reporting software is sensitive and can not afford to be shared with others data. However, there was no legal/compliance restrictions identified on the location of the storage of the data. The data can be moved to the cloud. Actually, the data for some of the customers, who consume the software solution hosted by DISYS in its datacenter, are in the DISYS datacenter already.

6.1.2. Model Services

With the limited documentation and the knowledge of employees, these services are also mapped into UML Models. The UML models are then inspected and manipulated to map the functionalities as services. The services are grouped based on business process they support, the product they support, the technology that is used to build them, the geographical domain that they can be run as well as based on the functional domain they support(e.g. accounting

department). We also use KDM (Knowledge discover meta-model) tools to represent the services in a platform independent manner.

The quality of service, the owner of the service, the security requirements of the service etc. are also identified and listed in the service directory. The dependence between the services is also modeled.

6.1.3. Model processes

Consequently, the associated business processes are modeled.

Its identified a part of the functionality of a business process can be deployed to cloud without having to re-write the whole tightly coupled legacy system. The consumers use accounting software that DISYS provides with a reporting software for generating the reports. The reporting software can be migrated to a cloud environment without disrupting the functionalities of the accounting software.

Mohagheghi et al. [61] have concluded that there was no way to map from the code to the business processes.

Hence, the current business process is modeled in detail manually. It's identified that the reporting software needs access to data that the accounting software produces. This dependency is shown by process modeling the system. Also, it's identified that it's not an internal process as often there was a need to integrate with 3rd party data. Hence, it's an ideal candidate to migrate to the cloud. Modeling the business process from the legacy system also allows us to re-engineer the business process. In our case, this business process model can be used to re-engineer the business process to host this system as a service. Hence, process modeling is very useful in transforming the IT as well as the business.

6.1.4. Model Governance

The existing governance model for changing the reporting software is modeled here. Essentially, the reporting software will need to be modified whenever any of the software in the portfolio that interacts with the reporting software changes its interface. This can occur as each client migrates its software across different versions. Also every change in the reporting software version needs to be manually installed on the client machine. The system support need to ensure compatibilities and the cost involved does not add value to the client as well as for DISYS.

This governance process is modeled in this sub-process.

6.1.5. Model Business Processes

In this migration, both the business and the IT are impacted. Hence, this sub-process might be expensive due to the effort involved. However, to migrate efficiently, this must be done in detail.

The current business process model of supporting the existing clients is modeled and the actors are identified.

DISYS sales team models the current sales contract model, while the support team models the current support model. The Sales team might also model the opportunity lost by not providing the software as a service to new customers.

The IT manager models the cost associated with the hardware, software and other entities responsible for the development of the software and support for the existing customers.

These models are used to compute the Key Performance Indicator (KPI) of the "as-is" state.

6.1.6. Model Rules

Existing Security and performance requirements are modeled by the IT personnel of DISYS. The privacy, compliance, legal requirements are modeled by the business experts. Other existing constraints are also modeled here.

6.1.7. Model Issues

Based on the models from the Model Business processes, the following issues are identified.

1. No resource sharing i.e. one installation of the software for every client using the reporting software.
2. Every update to the reporting solution needs to be upgraded on the client's run-time environment.
3. The report can not be consumed at the mobile phone of the client.
4. Potential new clients need to install the software to test its features and usage. They identify (i) Resource utilization, (ii) Maintenance Cost (different versions need to be maintained by development, sales and support teams) (iii) Business Reach(X% customer base increase) and (iv) Customer satisfaction (%) as the key performance indicators.

By defining this externally, the potential to migrate to a cloud model, where the software can be delivered as a service can be identified.

6.2. To-be Model

The to-be model for the target virtualized environment is built here.

6.2.1. Model Goals

Now, the new goals need to be modeled. After analyzing the current KPIs in the issues model, the following target KPIs are defined.

Maintenance cost will be cut by 10%, Resource utilization to be increased by 10%, and Pay-As-You-Go (PAYG) will need to be introduced. Now this requires some cost on migrating the legacy code. To make the initial decision, the SMART framework defined in [42] is applied to identify whether it makes sense to migrate this application to the cloud. The stakeholders then define the KPI for the enterprise based on the current as-is state.

6.2.2. Model Rules

1. New business rules for pay-as-you-model are defined.
2. Security and privacy requirements need to be modeled as the software is going to be hosted on cloud environment.
3. Procedure to update the reporting software to avoid any possible downtime needs to be developed.
4. There are no compliance/legal issues identified in migrating to the cloud.

6.2.3. Model Business processes

1. Delivery of report will be enabled via a browser.
2. Report can be consumed through mobile.
3. New business processes to enable Pay as you Go model has been modeled.
4. Integration with 3rd party data has been modeled.

6.2.4. Identify candidate processes, services and data

With no compliance and business restrictions from the business rules on the legacy reporting software, the whole software has been assigned as a candidate for cloud migration.

6.2.5. Model Actors/Roles

The sales as well as the support team is made aware of the strategy to migrate to cloud to start planning for new sales model while planning for decommission of the installed version of the software.

They also define the Quality of services that need to be provided on the cloud and are responsible to take corrective actions with the service consumers as well as with internal teams if need arises.

A Center of Excellence is formed to keep track of the migration issues which might be used in future cloud migration cycles. They are also responsible to monitor and to make sure that

the quality of service is met as agreed with the service consumers as well as with the cloud provider.

6.2.6. Plan testing

A new test plan needs to be drawn. On top of functional testing, several non-functional test cases are planned. For e.g. the issues like number of simultaneous requests, test-cases to snoop reports from another customers account etc.

Thus, the service level, process level, governance level, data-level testing, integration-level testing and security-level testing has been drawn.

6.2.7. Assigning Candidate data, services, processes for the clouds

Now, the to-be architecture is defined for the selected candidates taking into consideration of the new business architecture.

It has been defined using a modernized Microsoft SQL, Web Services and ASP .Net Components. Based on the to-be architecture platform, the components will need to be re-coded.

6.3. Change Process Model

6.3.1. Select target platform for each component

The platform selection for the reporting software is called to be performed.

Platform Selection

The candidates are listed:

Public cloud

Now, this tightly coupled application can be migrated to a public cloud. However, the data for the reporting software will need to be uploaded to the public cloud. Since a single client might need to update a few million rows, the scalability of data storage and the network bandwidth are a concern. Report querying is considerably light on network bandwidth. Since all the customers need to upload the data over the cloud, the costs and the network bandwidth are identified as potential bottleneck for public cloud migration.

Private cloud

Hence a private cloud is considered. In comparision with public cloud, the network bandwidth requirement is low as the accounting software is already hosted in the datacenter for many of the clients. Data replication is still an issue. However, it might be overcome once more and more software components are offered as a service and the data can be shared among services.

After analyzing the platforms, the software can be re-coded and tested on a private cloud as well as on a public cloud. The test results can be documented.

Select target platform.

Based on the test-results an appropriate cloud platform can be chosen. Its then continously monitored with gradual increase in traffic to make sure that it meets the required quality of service.

Next phases

Now, the present KPI is continously monitored over different cycles. If there are customer complaints, they are addressed. In case of persistent issues, the traffic on the target platform is reduced to fix the issues. Once the KPIs are showing an upward trend, migration can be increased.

After few cycles, the reporting software can be hosted as a service to clients and might see good resource utilization, increased customer base, cost saving from maintainence in addition to increased customer satisfaction.

The same process model can be used to migrate the accounting software going forward to further save costs and to increase revenue while becoming more efficient.

7. A South-Africa based Cellphone companys foray into Green IT

Lamb [60] discusses this case study on his paper about the use of GreenIT and Private Cloud in South Africa.

This use-case is for a large cellphone company in South Africa. IT has identified the following problems in their enterprise.

1. Test environment resource availability was a concern. There was no good scheduling process to share the test environment. There was considerable amount of wastage of servers. Every project that gets the server for testing did not release the server when testing was completed.

IT wanted to overcome this obstacle and here, we see how our process model can be applied in relation to the next steps of this migration effort. As defined in Architecture Driven Modernization (ADM) paradigm, this project involves a technical architecture transformation. Hence, the risk and the impact are low.

7.1. As-is Model

The existing **IT Infrastructure Library** (ITIL) tools are used to identify the current state of the datacenter.A section of the total servers are chosen in the problem domain.

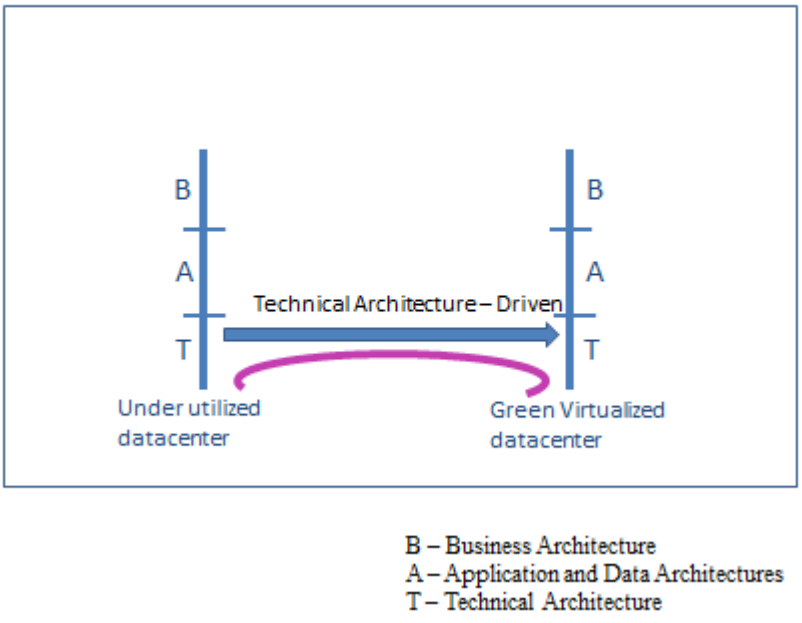


Figure 7-1 Green IT Migration

7.1.1. Model Data

- A data catalog is built around the problem domain, i.e. inefficient resource utilization. It has been found that the enterprise has more than 1100+ Server Instances in 2010.
- 40-60% of estimated spending is on maintaining current IT infrastructure.
- 12.97% is the average CPU usage across platforms – i.e.87.03% is the average CPU idle time. In test environment, it is even lower.
- 40% servers are estimated to be used in pre-production roles. These servers consume upto 40% of power and cooling (339,509 Watts).
- There are 231 End of Life Servers as of April, 2010. Of those 231, 149 Windows Servers with approx. 80 pre-prod, 44 AIX Servers with Approx. 25 Pre-prod and 38 Linux Servers. Approx. 20 Pre-Prod.

7.1.2. Model Services

These servers support a whole host of functionalities like enterprise web-applications, legacy systems, etc. to support functionalities (services) for different teams. These services need to be of the same state as it is on the source architecture, to support the same business processes today. Hence the services are modeled as a black-box to identify any issues that might arise due to the migration.
The service directory is built which defines the Quality of Service (QoS) of each of the services, the owner of the services, the security requirement of the services, etc.

7.1.3. Model Processes

The business processes that are being supported are being modeled here. This will help us to plan for the future state of these business processes. For e.g. the services that are being provided from this datacenter might need to be provided from another location. The dependent service consumers might be identified and intimated. Processes that might have potential problems due to some services relocation are also identified.
Also, the current business process is monitored to identify issues which can be re-engineered in the target platform.

7.1.4. Model Governance

The governance in the existing environment is modeled. The problem with the existing architecture is there is no proper governance in place to consume and to share resources. This has led to an improper mechanism to release server resources that were obtained for testing in projects. The current governance procedure is modeled here. On modeling, the lack of policies is identified as the major reason for poor resource utilization. The existing

governance model will be useful as a base to implement automated governance on the target platform.

7.1.5. Model Business Processes

Since it's a migration to a private cloud and is primarily driven by the IT, the degree of modeling effort required is considerably lower than when it's driven by business or when there is a change in the application and data architecture.

The current provisioning process is modeled here.

The IT provisioning/platform team manager models the current cost and resource utilization of the platform. He also models the time to provision a resource in the current environment. The IT semantics is modeled here as it's the driving force behind this migration effort.

There is little modeling effort on the business experts to ensure smooth cut-over to foresee any potential issues. The business semantics will continue to remain the same despite the migration.

7.1.6. Model Rules

The existing business rules are modeled. This helps us to identify if there will be any constraints, that one needs to be aware of on the new platform selection. Again, since it's an IT driven effort, the modeling effort is limited to IT. It's because the business rules are the same and modeling is expensive.

7.1.7. Model issues

The poor resource utilization and the maintenance costs are identified as the issues for the current state of the enterprise. Hence resource utilization (%) and maintenance costs ($$) are marked as KPI of the enterprise for the migration. This is decided after evaluating the models generated by the different departments in the actors/role model.

7.2. To-be Model

The to-be model for the target virtualized environment is built here.

7.2.1. Model Goals

Based on the KPI identified in the issues Model, the target values for the KPI factors resource utilization (%) and maintenance costs ($$) are identified.

A detailed spreadsheet is used to estimate the cost-savings and to make the decision for the migration. This cost-savings value is used to define the case for the budget of the migration.

7.2.2. Model Rules

The business rules are defined for the target to-be model. There was no change required in the business rules and hence a one-to-one mapping of the current architecture is done.

7.2.3. Model Business Processes

The new business process model is modeled. In this case, this will be almost again a one-to-one mapping from the as-is state.

7.2.4. Identify candidate processes, services and data

The rules model is checked to identify if some of the services/processes/data can not be moved to the target private cloud and no such restriction was found.

7.2.5. Model Actors/Roles

Since there is little/no impact on business, there is no need for business actors. In IT, a provisioning team is formed. They are responsible for any issues that may arise on the cloud platform. They are assigned their roles here.

7.2.6. Plan Testing

A detailed test plan to cover the functional and non-functional issues on the to-be system is defined here. This test plan covers all the domains of the architecture.

7.2.7. Assigning Candidate data, services, processes for the clouds

Based on the perfomance requirements in the test plan, if some of the services shouldn‘t be moved, it will be identified here. There will be no changes on the application or data architecture. In this case, the to-be technical architecture is a virtualized environment.

7.3. Change Process model

Here, the platform is the only component that needs to be migrated.

7.3.1. Select target Platform for each component

Platform selection

Private cloud is the target cloud type. One of the proposals was from IBM, which proposed its private cloud solution called CloudBurst.

CloudBurst will also be able to manage the external hardware as part of the cloud. So the existing hardware can also be used.

The resources are developed and deployed on the private cloud environment and the test plan is executed.

Similarly, the test results of other platforms are obtained.

They are all compared and IBM cloudburst is chosen as the target platform. Then, the to-be architecture is deployed. Its performance is monitored continuously.

Next Phases

Then, again the KPI of the system is checked in the new "As-is" state and further components are migrated to the target platform.

After migrating 280 developments and testing environments to the cloud, the number of servers has been reduced by 62(%) which equates to considerable maintenance cost savings. The governance process implemented has ensured good resource utilization.

The provision time for the testing environments has come down from 33 to 5.4 days (611%) while the test team members needed to provision the test environments has come down from 16 to 2 (800%).

8. Netflix‘s Transition to High-Availabilty Storage systems

This use-case is an example for the Information Technology Driven Application and Data Modernization where the source system is an Oracle RDBMS, while the target system is a highly available NoSQL storage.

Siddharth “Sid” Anand [90] discusses this case study on his paper about the Netflix’s Transition to High-Availability Storage Systems.

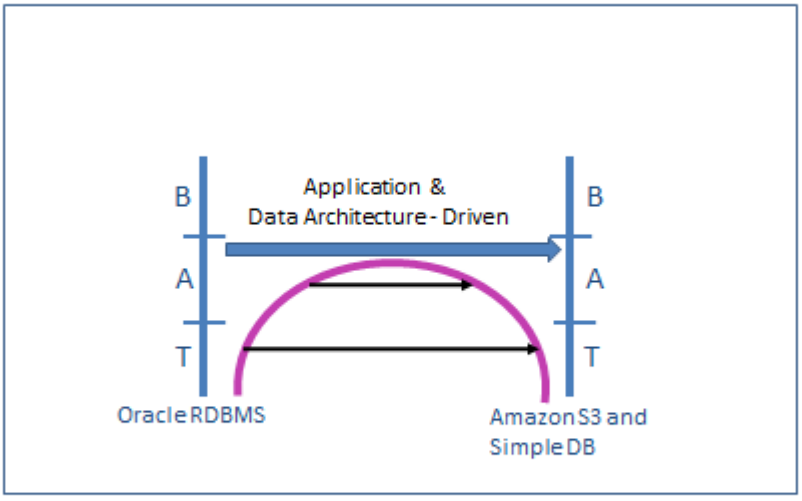


Figure 8-1 Netflix Data Migration

Netflix, Inc. is an American provider of on-demand Internet streaming media in the United States, Canada, Latin America, the Caribbean, United Kingdom, Ireland, Sweden, Denmark, Norway, Finland and flat rate DVD-by-mail in the United States. In April 2011, Netflix announced 23.6 million subscribers in the United States and over 26 million worldwide. By 2011, the total digital revenue for Netflix reached $1.5 billion [91].

By late 2008, Netflix had a single data center. This is a liability and it represents a single point-of-failure. With the rate of growth in both streaming adoption and subscription levels, this single data center will be insufficient – and they foresaw an immediate need for more power, better cooling, more space, and more hardware. One option was to build more data centers. This involves huge upfront costs and keeps IT away from its core competency: to deliver movies and TV shows.

Below we see how our process model can be applied to define the next steps in their modernization approach.

Since we propose an agile phased migration approach, a part of the product portfolio needs to be chosen as a problem domain. In our case, we choose to focus on Movie Recommendation infrastructure.

8.1. As-is Model

8.1.1. Model Data

The data catalog for the problem domain is built. Both MySQL and Oracle have been used to persist data. There were no privacy/security related issues with the data under consideration. Hence, there was no legal/compliance restrictions identified on the location of the storage of the data. The data can be moved to the cloud.

8.1.2. Model Services

Netflix's Information system is one which is built on Service-Oriented architecture (SOA). Hence, identifying services that are dependent on these data stores is fairly straight forward.

The quality of service, the owner of the service, the security requirements of the service etc. are also identified and listed in the service directory. The dependence between the services is also modeled.

8.1.3. Model processes

Consequently, the associated business processes are modeled. The processes involved in the domain are not internal processes and hence does not have any restrictions associated with the processes.

8.1.4. Model Governance

Leveraging on its existing SOA infrastructure, the policy artifacts involved are modeled and documented in this activity.

Because the existing architecture is service oriented, modeling the services, processes and governance are straight-forward.

8.1.5. Model Business Processes

Since this migration is primarily driven by the IT, the degree of modeling effort required is considerably lower than when it's driven by business. The higher-level business process is going to be the same and hence the involvement of business experts is limited.

The IT manager models the cost associated with the hardware, software and other entities responsible for the development of the software and support in the datacenter is built. The metrics on the number of requests for this movie recommendation services are identified and projected to find the future demand.

These models are used to compute the Key-Performance Indicators (KPI) of the "as-is" state.

8.1.6. Model Rules

Existing Security and performance requirements are modeled by the IT personnel of Netflix.

The privacy, compliance, legal requirements are modeled by the business experts.

Other existing constraints are also modeled here.

8.1.7. Model Issues

Based on the models from the Model Business Processes, the following issues are identified.

1. Scalability of the computing resources with respect to increasing demand is an issue.
2. Single point of failure needs to be avoided.
3. Data needs to be highly available.
4. The costs and the resource needed to build a new data center are estimated.

By defining this externally, the potential to migrate to a cloud model can be better identified.

8.2. To-be Model

The to-be model for the target environment is built here.

8.2.1. Model Goals

Now, the new goals need to be modeled. After analyzing the current KPI in the issues model, the following target KPI are defined.

Disaster recovery, managed fail-over, distribution, persistence, eventual consistency and support for a subset of SQL are identified as the new goals of the enterprise.

After evaluating the costs and the resource requirements to build and to maintain new datacenters, it has been identified as not one of the core-competencies of the organization.

8.2.2. Model rules

5. There were no compliance/legal/auditing issues that were identified in migrating to the cloud.
6. The business rules are not affected.
7. The IT rules with respect to resource provisioning and releasing have been modeled here.

8.2.3. Model Business processes

The high-level business process remains the same no-mater whether the data is stored on-premise or on the public cloud.

8.2.4. Identify candidate processes, services and data

With no compliance and business restrictions from the business rules, the data-stores are marked as candidates for migration to cloud.

8.2.5. Model Actors/Roles

The database administrators involved in the problem domain are assigned appropriate roles to prepare for the data replication. They identify and build the tools and the data catalog that need to be migrated to the cloud.

They have built an in-house framework known as IR(Item Replication) whose job is to replicate data from the data center to the public cloud.

8.2.6. Plan testing

Here, they plan and build tools for testing the different layers of the architecture, viz. the service level, process level, governance level, data-level testing, integration-level testing and security-level testing has been drawn.

8.2.7. Assigning Candidate data, services, processes for the clouds

Now, the to-be architecture is defined for the selected candidates taking into consideration of the new business architecture.

It has been defined using a NoSQL paradigm to replace the in-house RDBMS solution.

8.3. Change Process Model

8.3.1. Select target platform for each component

The platform selection for the problem domain is performed.

Platform Selection:

The candidates are listed. A large variety of choices based on NoSQL platforms have been listed. Amazon's Simple DB and S3 was one of them. Data has been migrated and tested based on the test plan.

Select target platform.

After analyzing the platforms, the Amazon's management of SimpleDB and S3 was a key adoption driver. Since SimpleDB's support for SQL-like languages appealed to their application developers, it has been chosen as the target platform.

Once chosen, a complete migration of the data-catalog is done by using their framework known as IR(Item Replication).

Next phases

Now, the present KPI are continously monitored over different cycles. If there are any issues, they are addressed. In case of persistent issues, the traffic on the target platform is reduced to fix the issues. Once the KPI's are showing an upward trend, migration can be increased.

After few cycles, netflix was able to rapidly migrate to the cloud while meeting aggressive product launch schedules and supprting high traffic growth.

9. Summary and Outlook

9.1. Summary

This master thesis proposes a generic, model-driven, platform independent process model that supports all the stakeholders to migrate existing applications into the cloud.

The existing solutions and best-practices of cloud migration were first investigated. After analyzing different case-studies, we identified Service Oriented Architecture as a strategy to reverse engineer the existing information systems by modeling them in terms of (i) the data they operate on, (ii) the services they provide to (iii) the business processes they support and (iv) the existing governance in the information system. These models can be used efficiently as base artifacts for building the architecture for the cloud platform.

It has been found from the different case-studies that most of the cloud migrations are driven by the IT to show value of cloud computing to the business by reducing cost, etc.. However, the scope of migration to cloud involves non-functional issues like privacy, security, etc. and these issues are not given its due diligence. To tackle this, our process model proposes that the business and the information technology experts model these non-functional requirements in a collaborative manner. Hence, the business processes and the business rules are modeled colloboratively and the Key Perfomance Indicator(KPI) of the existing architecture is then identified.

Based on the strategic business goals, target KPIs are defined in the "to-be" model. The business processes and the rules model developed earlier in the "as-is" model can be extended to define the corresponding models in the "to-be" model. Additionally, the actors involved to enforce these new rules and processes are identified in actors/roles model and a test-plan model is defined which provides the Quality of Service(QoS) and other non-fuctional requirements. Then, the "to-be" architecture is defined.

As standardization efforts like Semantic of Business Vocabulary and Business Rules (SBVR) [101] evolve, the rules and process models defined earlier can be (semi) automatically enforced in the information system implementation. Most of the organizations, specifically in the government, finance and health sectors need to maintain a high degree of compliance and privacy. Modeling these activities help to externalize these requirements while migrating to a cloud environment.

Once all these issues are externalized and the goals are modeled, cloud migration becomes that much more manageable and Information technology experts can focus on the technical aspects of the migration.

The new architecture is then deployed on candidate platforms to select a suitable one. The platform is monitored continously while decommissioning/reuse of existing infrastructure is planned. Then the KPI of the new "as-is" state of the architecture is evaluated and the above process can be repeated to further achieve Business-IT alignment.

It's been identified that Service oriented architecture leverages the cloud resources efficiently and hence the artifacts developed in the reverse engineering ("as-is") model can be used as a base to develop the corresponding to-be architecture. However, the process-model does not impose building a service oriented architecture solution as it's not always possible to adapt an existing information system to a SOA paradigm.

Three different types of real-world cloud migration scenarios have been shown to successfully fit into the proposed process model.

9.2. Outlook

To quote from the inner game of work [98],

P (Performance) = p (potential)-i (interference)

The interferences like Vendor lock-in, security concerns, etc. for embracing cloud Computing prevent enterprises to leverage on the full potential of cloud computing to maximize perfomance.

In order to utilize the "Web-as-a platform" to provide IT services and to fully leverage Cloud Computing, portability of cloud applications and services need to be ensured. It's here standards like TOSCA [96] that will further accelerate cloud computing adoption.

The potential of cloud computing to transform an organization's IT and business has been proven by many organizations [87]. One common denominator among all these organizations is that they embrace Service Oriented Architecture and this allowed them to leverage on the full potential of cloud computing. Embracing Service Oriented Architecture allows IT to change its application and data architecture semantics efficiently and respond to business requirements in an agile manner by leveraging cloud resources. Already, work has been going on to extend the SOA components to support cloud computing. For e.g. the author of this work has compared the composition engines to identify short-comings with respect to cloud computing [99]. Similarly, research has been going on for extending Enterprise Service Bus and other Service Oriented Architecture Components that will be needed to further adopt cloud computing.

Enterprise application portfolio consists of tightly coupled legacy systems. With the momentum continuing to accelerate in Software as a Service [94], [95] adoption, there is a need for traditional software vendors to provide their software as a service. Research has

been going on in projects like REMICS [100], where methods and tools are being developed for migrating these legacy systems to cloud platform. In order to achieve modernization of legacy systems, Architecture Driven Modernization [45] standards are being developed by OMG [35]. They need to be finalized and extended to support migration of these legacy systems to cloud platforms.

Appendix A

Enterprise Knowledge Discovery

Enterprise Knowledge Discovery (EKD) is a structured method to tackle ambiguous structured problems. Mostly, the result of an EKD application is documents that describe the problems and/or solutions to problems. Even if solutions are not found, the stakeholders involved understand the set of issues at hand. This means the EKD process produces knowledge in two forms; in the form of documents and in the form of knowledge in the minds of the people involved [1].

Application of EKD

Application of the Enterprise Knowledge Discovery mechanism results in the generation of Enterprise models. These enterprise models represent the various views of the enterprise. The various views are for example, the goals of the enterprise, impact of these goals on business processes, the constraints on these processes, the information system and the interfaces needed to realize these functionalities [1].

Broadly, an enterprise is visualized in the following 3 inter-connected Models

- Enterprise Goal Model
- Enterprise Business Process Model
- Enterprise Information System Model

Enterprise Goal Model

The goals of the enterprise are identified. This drives the business processes that must be developed to achieve the goals that are identified in this model.

This model helps an enterprise to identify its current goals, problems, cause, constraints and Opportunity [3].It also helps to identify the contradictions and relationships between those above items. It helps to agree upon a set of future goals from the various views of the enterprise. It is important to concentrate on the business itself, and not on the supporting information system and its more technical goals [4].

It is useful to use or at least to keep in mind some driving questions while doing goals modeling, e.g.:

- What are the strategies of this part of the enterprise?

- Are there stated policies in the enterprise that may influence this model?
- Which conventions, rules, regulations and laws are relevant?

Enterprise Business Process Model

This model represents the various aspects of enterprise processes. These processes are modeled based on the guiding principles which were defined by the Enterprise Goal model. In this model, in addition to the business processes to achieve the enterprise goals, all the actors and their roles are identified and a role/actor model is designed. People perform activities to achieve enterprise objectives. A role/activity model is designed that defines how resources are consumed / produced to achieve enterprise objectives. Activities carried out by defined roles deal with business objects. These objects are modeled into an Object Model. It defines the enterprise entities, attributes and relationships. The rules associated with each of the business processes are explicitly captured in a Business rules model.[4]

This is essentially a collaborative effort that helps to identify various aspects (like compliance, legal issues, privacy, performance requirements, governance) of the business processes. Once all these aspects are explicitly identified and captured, defining enterprise system model becomes that much easier.

Enterprise Information System Model

The enterprise business process model is used to model the information system model to perform the necessary business processes to reach the stated goals. This model is used as the requirements by the IT to build the information system.

Modeling using EKD

In enterprise knowledge modeling, different views of an enterprise are modeled which includes enterprise information systems, enterprise business processes and enterprise objectives. Since all the models are inter-related, the enterprise goals are more anchored across the different views of the enterprise.

References

1. CRI - Centre de Recherche en Informatique. The foundation of the EKD-CMM change management framework: EKD Method. http://crinfo.univ-paris1.fr/EKD-CMMRoadMap/foundationsEKD.html [last checked on 30th October 2012]

2. CRI - Centre de Recherche en Informatique. Understand Change Management within EKD-CMM.http://crinfo.univ-paris1.fr/EKD-CMMRoadMap/UnderstandingChangeManagementWithinEKD-CMM.htm [last checked on 30th October 2012]

3. CRI - Centre de Recherche en Informatique. Understand purpose of goal modelling. http://crinfo.univ-paris1.fr/EKD-CMMRoadMap/Understandpurposegoalsmodelling.htm [last checked on 30th October 2012]

4. Selmin NURCAN and Colette ROLLAND, Using EKD-CMM Electronic Guide Book for Managing Change in Organizations. http://crinfo.univ-paris1.fr/users/nurcan/pdf/EJ99camera.pdf [last checked on 30th October 2012]

5. Nesse, P.J.; Undheim, A.; Solsvik, F.H.; Dao, M.; Salant, E.; Lopez, J.M.; Elicegui, J.M.; , "Exploiting cloud computing — A proposed methodology for generating new business," Intelligence in Next Generation Networks (ICIN), 2011 15th International Conference on , vol., no., pp.241-246, 4-7 Oct. 2011 doi: 10.1109/ICIN.2011.6081083 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6081083&isnumber=6081051 [last checked on 30th October 2012]

6. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems, Volume 25, Issue 6, June 2009, Pages 599-616, ISSN 0167-739X, 10.1016/j.future.2008.12.001.
http://www.sciencedirect.com/science/article/pii/S0167739X08001957 [last checked on 30th October 2012]

7. Fehling, Christoph; Ewald, Thilo; Leymann, Frank; Pauly, Michael; Rütschlin, Jochen; Schumm, David, Capturing Cloud Computing Knowledge and Experience in Patterns, http://www.iaas.uni-stuttgart.de/institut/mitarbeiter/fehling/INPROC-2012-

22%20Capturing%20Cloud%20Computing%20Knowledge%20and%20Experience%20in%20Patterns.pdf [last checked on 5th November 2012]

8. *ZDNet. Coghead's demise highlights PaaS lock-out risk.* http://www.zdnet.com/blog/saas/cogheads-demise-highlights-paas-lock-out-risk/668 [last checked on 5th November 2012]

9. TechCrunch. Update: Amazon Web Services Down In North Virginia — Reddit, Pinterest, Airbnb, Foursquare, Minecraft And Others Affected. http://techcrunch.com/2012/10/22/aws-ec2-issues-in-north-virginia-affect-heroku-reddit-and-others-heroku-still-down/ [last checked on 5th November 2012]

10. InformationWeek. Inside Zynga's Big Move to Private Cloud. http://www.informationweek.com/hardware/virtualization/inside-zyngas-big-move-to-private-cloud/232601065 [last checked on 5th November 2012]

11. IBM. SOA for Dummies. http://www-01.ibm.com/software/solutions/soa/offers/soafordummies/ [last checked on 5th November 2012]

12. Keith Brown, M. R. Pamidi, Ph. D. CloudBeat 2011. http://www.itnewswire.us/CloudBeat_2011.pdf [last checked on 5th November 2012]

13. InformationWeek. Google App Engine Price Hike Stuns Developers. http://www.informationweek.com/cloud-computing/platform/google-app-engine-price-hike-stuns-devel/231600672 [last checked on 5th November 2012]

14. Barrett, J.L. (1994) Process visualization, Getting the vision right is key, *Information Systems Management*, spring 1994. [last checked on 8th November 2012]

15. Janis Bubenko jr,, Anne Persson, Janis Stirna. EKD User Guide. ftp://ftp.dsv.su.se/users/js/ekd_user_guide_2001.pdf [last checked on 8th November 2012]

16. S. Shunmuga Krishnan , Ramesh K. Sitaraman, Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs. http://people.cs.umass.edu/~ramesh/Site/HOME_files/imc208-krishnan.pdf [last checked on 8th November 2012]

17. TechTarget. http://searchsoa.techtarget.com/feature/Business-architecture-takes-the-fore-in-application-portfolio-analysis [last checked on 8th November 2012]

18. Maximilien Kintz, A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology. http://www.e-business.iao.fraunhofer.de/Images/MODIQUITOUS%202012%20_%20CR2_tcm462-104922.pdf [last checked on 8th November 2012]

19. David S.Linthicum, Cloud Computing and SOA Convergence in Your Enterprise. Addison-Wesley Information Technology Series, 2010

20. Janis Stirna, Anne Persson. Ten Years Plus with EKD: Reflections from Using an Enterprise Modeling Method in Practice. http://ceur-ws.org/Vol-365/paper10.pdf [last checked on 8th November 2012]

21. SANNI Lookman, Extending Chicken Little strategy in legacy system migration, http://lookouster.org/blog/2010/10/14/201/ [last checked on 8th November 2012]

22. F.Leymann, Skript, “Loose Coupling Lecture”, 2012

23. F.Leymann, Skript, “Service Computing Lecture”, 2012

24. F.Leymann, Skript, “Business Process Management Lecture”, 2012

25. Tom Philip and Erik Wend. Early Warning Signs of Failure in Offshore Outsourced Software Project – An Indo-German Case Study. http://www.ifi.uzh.ch/pax/uploads/pdf/publication/1626/EWS_of_Failures_in_OSD_Project_Indo_German_Case_Study.pdf [last checked on 10th November 2012]

26. IBM. Develop a migration strategy from a legacy enterprise IT infrastructure to an SOA-based enterprise architecture. http://www.ibm.com/developerworks/webservices/library/ws-migrate2soa/ [last checked on 10th November 2012]

27. T.G.J. Schepers, M.E. Iacob, P.A.T. Van Eck, A lifecycle approach to SOA governance http://doc.utwente.nl/64697/1/Schepers_SAC2008_final.pdf [last checked on 10th November 2012]

28. Dennis Smith, Migration to Service Oriented Architecture (SOA) with Selected Research Challenges http://cgi.cse.unsw.edu.au/~soc/icsoc08/summer_school/soa_migration.pdf [last checked on 30th October 2012]

29. Kdmanalytics. Semantic of Business Vocabulary and Business Rules. http://kdmanalytics.com/sbvr/index.php [last checked on 30th October 2012]

30. Netflix. Chaos Monkey Released Into The Wild. http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html [last checked on 10th November 2012]

31. T.G.J. Schepers, M.E. Iacob , P.A.T. Van Eck, A lifecycle approach to SOA governance, http://doc.utwente.nl/64697/1/Schepers_SAC2008_final.pdf [last checked on 10th November 2012]

32. Erl, T, Service Oriented Architecture: Concepts, technology and design, Prentice Hall, Upper Saddle River, NJ, 2005.

33. Techtarget. SOA policy management. http://searchsoa.techtarget.com/answer/SOA-policy-management [last checked on 10th November 2012]

34. Colette Rolland, REASONING WITH GOALS TO ENGINEER REQUIREMENTS, http://hal.archives-ouvertes.fr/docs/00/70/64/94/PDF/ICEISinvitedpaper2003-Rolland.pdf [last checked on 10th November 2012]

35. OMG. OMG Model Driven Architecture. http://www.omg.org/mda/ [last checked on 10th November 2012]

36. Certificate of Cloud Security Knowledge. Cloud Adoption from the Bottom Up. http://ccskguide.org/cloud-adoption-from-the-bottom-up/ [last checked on 10th November 2012]

37. Hinkelmann , Enterprise Architecture – Introduction Business-IT Alignment and Agility http://knut.hinkelmann.ch/lectures/ea/EA_1_Introduction_Alignment_Agility.pdf [last checked on 10th November 2012]

38. Intel. SOA Service Governance Policy Enforcement. http://cloudsecurity.intel.com/solutions/soa-web-service-governance [last checked on 10th November 2012]

39. S-CUBE consortium. Survey on Business Process Management. http://www.s-cube-network.eu/results/deliverables/wp-jra-2.1/PO-JRA-2.1.1-State-of-the-art-survey-on-business-process-modelling-and-Management.pdf/at_download/file [last checked on 10th November 2012]

40. David Hollingsworth, The workflow reference model. David Hollingsworth, Winchester Workflow Management Coalition 1995, http://www.wfmc.org/standards/docs/tc003v11.pdf [last checked on 10th November 2012]

41. Frank Leymann, Dieter Roller, Production workflow: concepts and techniques, 2000, http://www.amazon.com/Production-Workflow-Techniques-Frank-Leymann/dp/0130217530 [last checked on 10th November 2012]

42. Carnegie Mellon. SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment http://www.sei.cmu.edu/library/abstracts/reports/08tn008.cfm [last checked on 10th November 2012]

43. Chung, S., Young, P., & Nelson, J. "Service-Oriented Software Reengineering: Bertie3 as Web Services." *Proceedings of the 2005 IEEE International Conference on Web Services (ICWS'05.* Orlando, FL (USA), July 11−15, 2005. IEEE Computer Society, 2005. Digital Object Identifier 10.1109/ICWS.2005.109.

44. CISCO. Cloud computing in the Public Sector: Public Manager's Guide to Evaluating and Adopting Cloud Computing. http://www.cisco.com/web/about/ac79/docs/wp/ps/Cloud_Computing_112309_FINAL.pdf
[last checked on 10th November 2012]

45. OMG. Architecture-Driven Modernization 101: Concepts, Strategies & Justification http://www.omg.org/news/meetings/workshops/ADM_2005_Proceedings_FINAL/T-1_Ulrich.pdf [last checked on 10th November 2012]

46. IBM. IBM Business Process Manager Advanced. http://www-01.ibm.com/software/integration/business-process-manager/advanced/features/ [last checked on 10th November 2012]

47. Dr. Vitaly Khusidman, William Ulrich, Architecture-Driven Modernization: Transforming the Enterprise, http://www.omg.org/cgi-bin/doc?admtf/07-12-01.pdf [last checked on 10th November 2012]

48. C. Ward, N. Aravamudan, K. Bhattacharya, K. Cheng, R. Filepp, R. Kearney, B. Peterson, L. Shwartz, C. C. Young , Workload Migration into Clouds – Challenges, Experiences, Opportunities, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5557998 [last checked on 10th November 2012]

49. Rashmi, Dr.Shabana Mehfuz, Dr.G.Sahoo, A five-phased approach for the cloud migration, http://www.ijetae.com/files/Volume2Issue4/IJETAE_0412_48.pdf [last checked on 10th November 2012]

50. Cloud Security Alliance .Security guidance for critical areas of focus in cloud computing. https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf [last checked on 10th November 2012]

51. Lookman SANNI, Master Thesis "Extending Chicken Little methodology in legacy system migration", [last checked on 10th November 2012]

52. Michael L. Broadie, Michael StoneBraker, DARWIN: On the Incremental Migration of Legacy Information Systems, http://db.cs.berkeley.edu/papers/S2K-93-25.pdf [last checked on 10th November 2012]

53. Jesus Bisbal, Deirdre Lawless, Bing Wu, Jane Grimson, Legacy Information System Migration: A Brief Review of Problems, solutions and Research Issues

54. Ian Warren and Jane Ransom, Renaissance: A Method to Support Software System Evolution, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1045037&contentType=Conference+Publications&queryText%3DRenaissance%3A+A+Method+to+Support+Software+System+Evolution [last checked on 10th November 2012]

55. Warren. I.(ed)-2000, The renaissance of Legacy systems.

56. N. Ganti, W. Brayman - John Wiley & Sons Inc., 1995, Transition of Legacy Systems to a Distributed Architecture.

57. Stephen Kaisler, William H. Money, Stephen J. Cohen, Decision Framework for Cloud Computing, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6149073 [last checked on 10th November 2012]

58. Markus Klems, Jens Nimis, Stefan Tai, Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing, http://cyberaide.googlecode.com/svn/trunk/misc/cloud-papers/cca08/12.pdf [last checked on 10th November 2012]

59. Hugo Brunetiere, Jordi Cabot, Frédéric Jouault, MoDisco: A Generic And Extensible Framework For Model Driven Reverse Engineering [last checked on 10th November 2012]

60. John Lamb, Green IT and use of Private Cloud Computing in South Africa, [last checked on 10th November 2012]

61. Parastoo Mohagheghi, Thor Saether, Software Engineering Challenges for Migration to the Service Cloud Paradigm,

62. Andrikopoulos, Vasilios, Fehling, Christoph, Leymann, Frank, Designing for CAP - The Effect of Design Decisions on the CAP Properties of Cloud-native Applications, http://www.iaas.uni-stuttgart.de/RUS-data/INPROC-2012-09%20-%20Designing%20for%20CAP.pdf [last checked on 10th November 2012]

63. OMG. Model-Driven Architecture: Vision, Standards And Emerging Technologies. http://www.omg.org/mda/mda_files/Model-Driven_Architecture.pdf [last checked on 10th November 2012]

64. National Institute of Standards and Technology. The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [last checked on 10th November 2012]

65. Techtarget. Business Process Management. http://searchcio.techtarget.com/definition/business-process-management [last checked on 10th November 2012]

66. GCN. Consolidation at the crossroads: 4 paths taken. http://gcn.com/articles/2012/04/02/email-consolidation-at-the-crossroads.aspx [last checked on 10th November 2012]

67. Informationweek. GSA Moving USA.gov, Data.gov To Public Cloud. http://www.informationweek.com/government/cloud-saas/gsa-moving-usagov-datagov-to-public-clou/232500473 [last checked on 10th November 2012]

68. Bundesministeriums für Wirtschaft und Technologie. Energieeffiziente IKT für Mittelstand,Verwaltung und Wohnen – IT2Green. http://it2green.de/documents/20120224_IT2GREEN_Broschuere_barrierefrei.pdf [last checked on 10th November 2012]

69. GAMES. Green Active Management of Energy in IT Service Centers. http://www.green-datacenters.eu/ [last checked on 10th November 2012]

70. PCWorld. Gmail Outage Marks Sixth Downtime in Eight Months. http://www.pcworld.com/article/160153/google.html [last checked on 10th November 2012]

71. CNET. Amazon cloud outage impacts Reddit, Airbnb, Flipboard. http://news.cnet.com/8301-1023_3-57537499-93/amazon-cloud-outage-impacts-reddit-airbnb-flipboard/ [last checked on 10th November 2012]

72. Legacy Systems Migration - A Method and its Tool-kit Framework, Bing Wu, Deirdre Lawless, Jesus Bisbal, Jane Grimson, Vincent Wade, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=640188 [last checked on 10th November 2012]

73. M. Klems, J. Nimis, and S. Tai, "Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing" Markets, Services, and Networks, Lecture Notes in Business Information Processing, vol. 22, 2009.

74. OMG. Knowledge Discovery Metamodel. http://www.omg.org/technology/kdm/index.htm [last checked on 10th November 2012]

75. Wikipedia. Service Oriented Architecture. http://en.wikipedia.org/wiki/Service-oriented_architecture [last checked on 25th November 2012]

76. Mohammed A Aboulsamh, Towards a model-driven approach for planning a standard-based migration of enterprise applications to SOA, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5190631 [last checked on 25th November 2012]

77. K. Magoutis, M. V. Devarakonda, N. Joukov, N. G.Vogl, “Galapagos: Model-driven discovery of end-to-end application - storage relationships in distributed

systems. IBM Journal of Research and Development, Vol. 52, No. 4-5, 2008, Pages 367-378.

78. C. Ward, V. Aggarwal, M. Buco, E. Olsson, S.Weinberger, “Integrated Change and Configuration Management”, IBM Systems Journal, Vol. 46, No. 3, July 2007, Pages: 459-478.

79. Harry M. Sneed, Wrapping Legacy Software for Reuse in a SOA, Third GI-Workshop XML Integration and Transformation for Business Process Management, Passau (Germany), 22 February 2006.

80. Michele Cantara, Common Features of External Service Providers' SOA Frameworks and Offerings, publication date, 26 September 2005.

81. Ali Khajeh-Hosseini, David Greenwood, Ian Sommerville, Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5557962&contentType=Conference+Publications&searchField%3DSearch_All%26queryText%3DCloud+Migration%3A+A+Case+Study+of+Migrating+an+Enterprise+IT+System+to+IaaS [last checked on 25th November 2012]

82. HyperStratus. Migrating Applications to the Cloud An Amazon Web Services Case Study. http://hyperstratus.com/migrating-applications-cloud [last checked on 25th November 2012]

83. Meiko Jensen, J¨org Schwenk, Nils Gruschka, Luigi Lo Iacono, On Technical Security Issues in Cloud Computing http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5284165

84. M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing.," 2009.

85. Nextgov. NOAA MOVES TO CLOUD-BASED EMAIL. http://www.nextgov.com/cloud-computing/2012/01/noaa-moves-to-cloud-based-email/50396/ [last checked on 10th November 2012]

86. Techtarget. Unix to Linux migration: Five Gartner tips for Indian businesses. http://searchdatacenter.techtarget.in/tip/Unix-to-Linux-migration-Five-Gartner-tips-for-Indian-businesses [last checked on 10th November 2012]

87. Highscalability. Startups Are Creating A New System Of The World For IT. http://highscalability.squarespace.com/blog/2012/5/7/startups-are-creating-a-new-system-of-the-world-for-it.html [last checked on 10th November 2012]

88. Infoworld. The 10 worst cloud outages (and what we can learn from them), http://www.infoworld.com/d/cloud-computing/the-10-worst-cloud-outages-and-what-we-can-learn-them-902 [last checked on 10th November 2012]

89. Vivek Kundra, FEDERAL CLOUD COMPUTING STRATEGY, http://www.mail.governmenttrainingcourses.net/pdfs/Federal-Cloud-Computing-Strategy1.pdf [last checked on 10th November 2012]

90. Siddharth "Sid" Anand, Netflix's Transition to High-Availability Storage Systems, https://sites.google.com/site/practicalcloudcomputing/index/Netflix%E2%80%99sTransitiontoaKey_v3.1.pdf?attredirects=0 [last checked on 10th November 2012]

91. Wikipedia. Netflix. http://en.wikipedia.org/wiki/Netflix [last checked on 10th November 2012]

92. Netflix. Announcing Blitz4j - a scalable logging framework. http://techblog.netflix.com/2012/11/announcing-bitz4j-scalable-logging.html [last checked on 10th November 2012]

93. C ouvade. IDC Prediction: SaaS Revenue Growth 6X Faster Than All Software.http://www.cloudave.com/6785/idc-prediction-saas-revenue-growth-6x-faster-than-all-software/ [last checked on 10th November 2012]

94. Couvade. IDC Prediction: SaaS Revenue Growth 6X Faster Than All Software. http://www.cloudave.com/6785/idc-prediction-saas-revenue-growth-6x-faster-than-all-software/ [last checked on 10th November 2012]

95. Gartner. Gartner Says Worldwide Software-as-a-Service Revenue to Reach $14.5 Billion in 2012. http://www.gartner.com/it/page.jsp?id=1963815 [last checked on 10th November 2012]

96. OASIS. OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC. https://www.oasis-open.org/committees/tosca/ [last checked on 10th November 2012]

97. 4CaaSt. Welcome to 4CaaSt. http://www.4caast.eu/ [last checked on 10th November 2012]

98. Theinnergame. The Inner Game of Work. http://theinnergame.com/products/books/the-inner-game-of-work/ [last checked on 10th November 2012]

99. Thomas Bachmann, Lukasz Bialy, Anand Babu, Comparison of Composition Engines and Identification of Shortcomings with Respect to Cloud computing., http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=FACH-0146&mod=0&engl=0&inst=IAAS [last checked on 10th November 2012]

100. REMICS. Reuse and Migration of legacy applications to Interoperable Cloud Services. http://www.remics.eu/ [last checked on 10th November 2012]

101. OMG. Semantics Of Business Vocabulary And Business Rules. http://www.omg.org/spec/SBVR/1.0/ [last checked on 10th November 2012]

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

Declaration for the Master's Thesis

I warrant, that the thesis is my original work and that I have not received outside assistance.

Only the sources cited have been used in this draft. Parts that are direct quotes or

Paraphrases are identified as such.

_____ _____

(Anand Babu Ramalingam)