2016

Senato della Repubblica

Ing. Vincenzo Curci
Certified Itil Expert

[ITIL SYSTEM METHODOLOGY]

V1.1

An ITIL project history: pretending to be a methodology to manage and design an ITIL project

1 Summary

2 Foreword

This booklet is the sum of experiences, gathered till now, during the (on-going) development and deployment of an ICT SKMS and (small) ERP, adopting the ITIL philosophy.

The Project, as stated, is still on going, because ITIL is a as useful as wide: the real IT service management world is at least as complex as the ITIL theory.

Introducing ITIL in a provider organization is a daunting task; inserting a new tool to achieve this, makes it even more complex.

However, till now we can say we have been quite successful, even if with a significant effort (and lot of problems to solve, but this is our work, isn't it?).

The approach described in the booklet is obviously not the only approach to ITIL, but optimistically it could result useful in other medium small organizations like in our case.

I hope that the journey to ITIL, told in this booklet, can give some helpful suggestions to others, who have the determination and/or the need to embrace ITIL and some tools supporting it. My best hope is that everybody, interested in ITIL and system development, can find something useful in it.

The booklet is structured in the following way:

From Chapters 3 to 6, the ITIL approach, the platform and the context of the project are described.

In Chapter 7, the overall project organization is explained.

From Chapters 8 to 9, the Concept and Data dimension is dealt with, with hints about the realization of the CMDB.

From Chapters 10 to 13, the meaning of Activity Domain is explained and the approach to process analysis is described in detail.

Chapter 14 is about reports and measures, from the data gathered in the system.

Chapter 15 is about external system integration.

Chapters 16 -17 give some details on activities about project quality, like Style rules and testing

Chapter 18 describes the approach adopted for supplier management

Chapter 19 is about the details on the templates developed for process analysis, testing and other project activities.

The booklet is **CMDBuild** oriented, as this platform was our choice and many ideas and practices are definitively linked to the platform.

3 Let's start: Reference Library ITIL

The ICT department decided (more precisely its boss and part of the Ict management, with an harder to convince part of the rest of Ict management) to start applying a more structured approach to ICT operations and life-cycles.

The question was: which guidelines or best practices use as a reference?

With its presence on the market now for a long time, the choice was about simple: ITIL (v3).

As a comprehensive organized set of best practices, its view based on the user/customer needs and on the services delivered to the user/customer, ITIL, in my opinion, can be considered one of the best references about the internal organization and management of a ICT service provider, be a company or an internal department.

Now the Ict Department of Italian Senato della Repubblica is heading towards a progressive ITIL implementation, not easy to accomplish and still a work in progress. The ITIL culture is now part of the organization: skills and competences are developed and framed along ITIL guidelines.

The main important aspect is to "start the journey", that is to make things move, to start and keep going the (slow) change of mind of all ICT people, to adapt and adopt this new frame.

Key term adoption of ITIL terminology and definitions is one critical success factor to start change. The **adoption of an automation platform with a CMDB** (to implement ITIL processes) is another critical success factor.

4 Methodological Approach

Organizational Approach

The adoption of **ITIL Best Practices** is quite **complex** for two main aspects: **organizational** and **technical**.

The greatest difficulties are, imho, in the organizational field. When I think about organization, I think about people and their "habits" on the workplace (call it organizational culture or whatsoever).

Even if the concepts, described in ITIL, are already present in the service provider organizational structure, normally they are not formalized or documented. Perhaps the organization developed internal best practices, which proved useful to solve problems, but there is few consciousness, if any, of them.

The organizational system is intended, at most, as the organizational structure diagram (even at an high level), with the only hierarchical relationships and, at best, generic job description. Every structure has a light description of high level tasks. Things normally go on without many structured/documented processes ("this is the way we always made and it works, so what"). The existence of organizational substructures is sometimes recognized, but only when a problem or a particular need arises.

The main tool of internal interaction is the email, with the major sophistication of mail groups, which normally correspond with the "real" organizational substructures, that is those substructures which are not documented (too much detail!) on any org chart, but are those which carry on the work.

So the first approach towards the introduction of ITIL best practices is a preliminary phase of organizational analysis, to understand the **real** activity organization, in the sense of who makes what (RACI analysis), in which order (Process) and how this maps on ITIL, if any.

The ITIL core is huge and can be overwhelming. It has to be adapted to the inner workings of the service provider, who existed well before ITIL introduction and worked so far, even if not necessarily at its best.

The approach to an ITIL introduction is not a single one, as it depends on many factors linked to the actual provider situation:

- How well (or bad) the provider is already structured
- How big is the structure

- How change-oriented is the structure
- Where are the main problems to be addressed at the moment of the analysis
- Where the work/service for users/customers has bigger impact at the moment
- What is the strategy or the direction held by the top management

So the right approach can be condensed in a statement: it depends.

Adopting a top down approach to cover all of the ITIL body in one big step is not necessarily the best approach.

The aim is to cover progressively the Library, using an opportunistic approach to select, one step at the time, the organizational boundaries to be "ITILised". These organizational "boundaries" can be defined again through many criteria, which depend on the provider internal situation. The "boundaries" are functional only to the plan of progressive ITILization of the provider, they don't necessarily match with an organizational substructure or define a new one, e.g. a boundary could cover a set of processes cross many org structures.

The organizational change is in this case iterative and subdivided in chunks, that can be more easily addressed.

However, **some global starting steps** are useful, to understand the provider overall organizational context, before starting to identify specific boundaries and go deep on them:

- **An initial analysis of the overall organizational structure**, in term of formal hierarchical diagram and competence of every organizational unit
- An quick check of **how the main ITIL processes are currently (if any) addressed**, more or less formally
- A review of **the main tools adopted at the purpose**, if any, and the level of operators satisfaction and domain coverage (how good is the tool to address the need/process domain).

The results of these steps is a sort of rough map of the situation, a reasonable insight of the organization (someone would say an assessment, but the term implies an heavyweight and costly work, which immediately dooms ITIL implementation to a premature death). This map gives indications of the main gaps to address.

An high level roadmap can be traced on the prioritized needs and gaps to address, to divide the overall context in many smaller domains (the "boundaries"), to be singularly approached. **The domain is defined as a subset of ITIL concepts,** that make sense to address in a single project, in a specific provider, and (again!) **depends** on the specific provider at a specific time.

NOTE: Needs and priorities keep changing as time passes, even in the same provider, because the world is constantly changing. So, if the map you discovered and the consequent roadmap can be useful

in a certain moment, after a reasonable amount of time (could be a year, but even a bunch of months) they could be outdated, even partially. In this latter case, a refocus of the priorities can be necessary.

Then more in-depth steps can be undertaken, to redefine and redesign organization and processes for every domain.

Even in these following steps, it's difficult to address all the requirements, so the advice is to follow an iterative approach, to improve and refine the achieved results, to even better align the new organizational system to the real needs of the provider.

Iterating on a domain in a cyclic (agile) approach, to release a new version of the organizational system on the same domain, or start approaching a new domain, is always a question of opportunity. A roadmap can be followed to be sure to cover, at the end, all the aspect of ITIL, but contingency and events can change the order of the domains in the roadmap.

Technical approach

The number of processes, artifacts and verification points in ITIL can be huge and not easy to maintain once in production.

Adopting a new formalized process without the support of an automation tools, in the experience of the writer, is almost useless and sometimes dangerous.

In the better case, all the set of steps and artifacts is perceived as a useless waste of time; in the worst the risk of over bureaucracy is behind the corner.

If a process must be followed, this process must be carefully designed, documenting all the relevant information, remediation procedures, artifacts and the involved configuration items, and then implemented on an automation system, backed by a sound CMDB.

Only an automation system can guarantee that every process instance is made always in the same way, collecting always all the relevant information and involving all the right actors. The trace on the system makes possible and easier both post-mortem analysis and SLA management.

So the main points for the technical approach are:

- Select an **automation tool (an ICT ERP)**
- Adopt the **right project management attitude**

- Work on a **carefully designed CMDB**

5 Characteristics of the Automation Tool

To define the characteristics of the "best" automation tool for ITIL is not an easy task.

Perhaps the "best" tool, in absolute terms, doesn't exist at all.

Many (costly) tools on the market, that boasts ITIL full compliance and coverage, are quite a huge mouthful.

They are often **designed to manage the most complex case in the most complex configuration** of everything, with the most complex processes perhaps only useful in really big companies, with lots of organizational structures and people.

This approach has many **drawbacks**, imho, at least in a smaller organization:

- **Work for difference:** You make the analysis of your environment (CMDB and processes) to try to adapt the tool to your needs, removing stuff in a sort of differential compromise, but you must analyze the tool to make it well too. This is a daunting task, if the tool is so complex. The analysis effort is almost double (or more): the tool and your organization. At the end, this kind of tools tend to impose their organizational model on your environment and this is not necessarily a good thing.
- **Customization will stay**: Nevertheless you'll need also customizations, because a tool, as complete as it can be, will always lack something you need. New types of Configuration Items, to model an ever changing technology, can be a driver to make further customizations.
- **Tool constraints**: So you need to move among many constraints to adapt the tool to your organization and vice versa.
- **Locked**: At the end you could be locked with a costly supplier.

However, if it happens that the tool covers almost exactly all your needs and mimics your organization perfectly, that's the tool for you (I'm never been so lucky).

Normally, we are not so fortunate and we already have a set of requirements, which these big tools sometimes don't cover adequately in all their complexity.

A **radically different approach** is to select **a tool that is almost "empty"**, that is, it doesn't implement a specific organizational model or CMDB model, (or perhaps a very light one), but let you define your own in every detail in a playground of basic services.

Tools like these offer a completely configurable CMDB or even DB to model your Configuration Items and every other concept of the SKMS, with a strong foundation of platform services like:

- user and privilege management in a very granular way,
- relationship definition,
- domain value definition,
- report generation,
- dashboard definition,
- workflow engine integrated with the CMDB
- and a standard uniform GUI.
- ...

This set of features is not linked to a specific CMDB or process model: you can arrange yours, based on your effective needs, even in an incremental way. This approach needs a careful analysis, as we'll see in next chapters, but give a result that is tailored on your requirements (fit for use AND fit for purpose). No more, no less.

Our choice has been **CMDBuild**, an open source platform, which is built to purpose without a predefined CMDB model, has a powerful workflow engine, well integrated with the data to manage and is completely configurable.

Another advantage of such a kind of tool lays in its inherent plasticity: as world changes, ICT changes too and very fast, so it is easier to keep the tool up to date to emerging needs.

6 Roadmap definition

A possible roadmap, the one adopted in this project, is visualized in the following diagrams.

The reference schema is the ITIL diagram of the SKMS (Service Knowledge Management System). This diagram has been revisited to specific project needs. Over this diagram some areas are designed. Every area covers one or more domains of an iteration in roadmap. In some iterations, some new modules can appear, detailing or even modifying the scope of the project, according to user priorities and requirements.

The representation is quite high level and can be further detailed with the list of features. However this visual approach can be easily grabbed also by the non technical or managerial stakeolder, to make clear which is the vision of the project, which are the objectives, how to get to them and through which steps on a timeline.

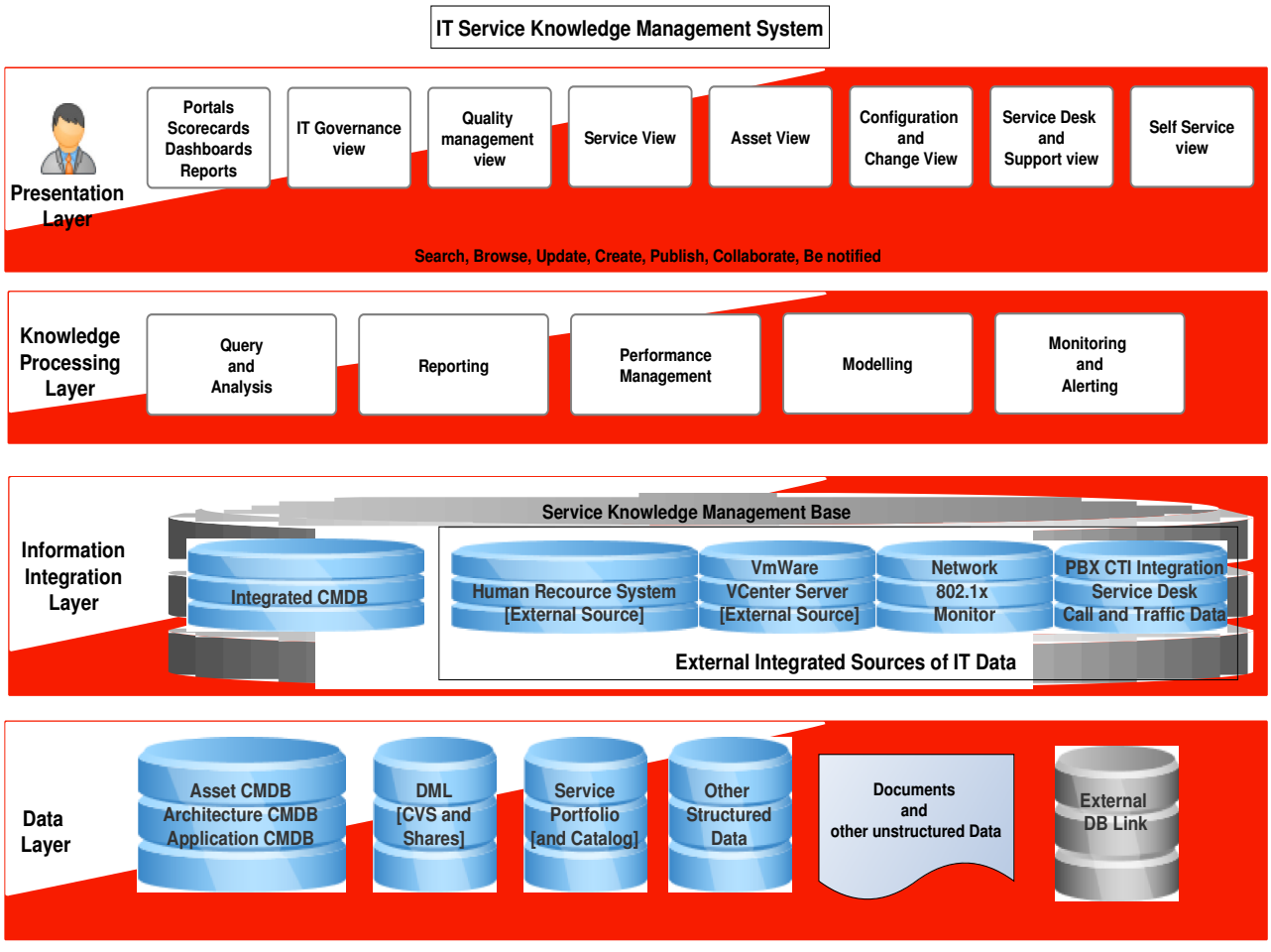


Figura 1 CMDBuild Senato: Target vision

Version 1.0

Version 1.0 is the big start of the project. As it usually happens, the first domains to approach are the gold couple **Asset Management** and **Service Desk**. In our case there was also a tool to replace, already covering most of these domains. They are the most critical areas as they directly involve the end user.

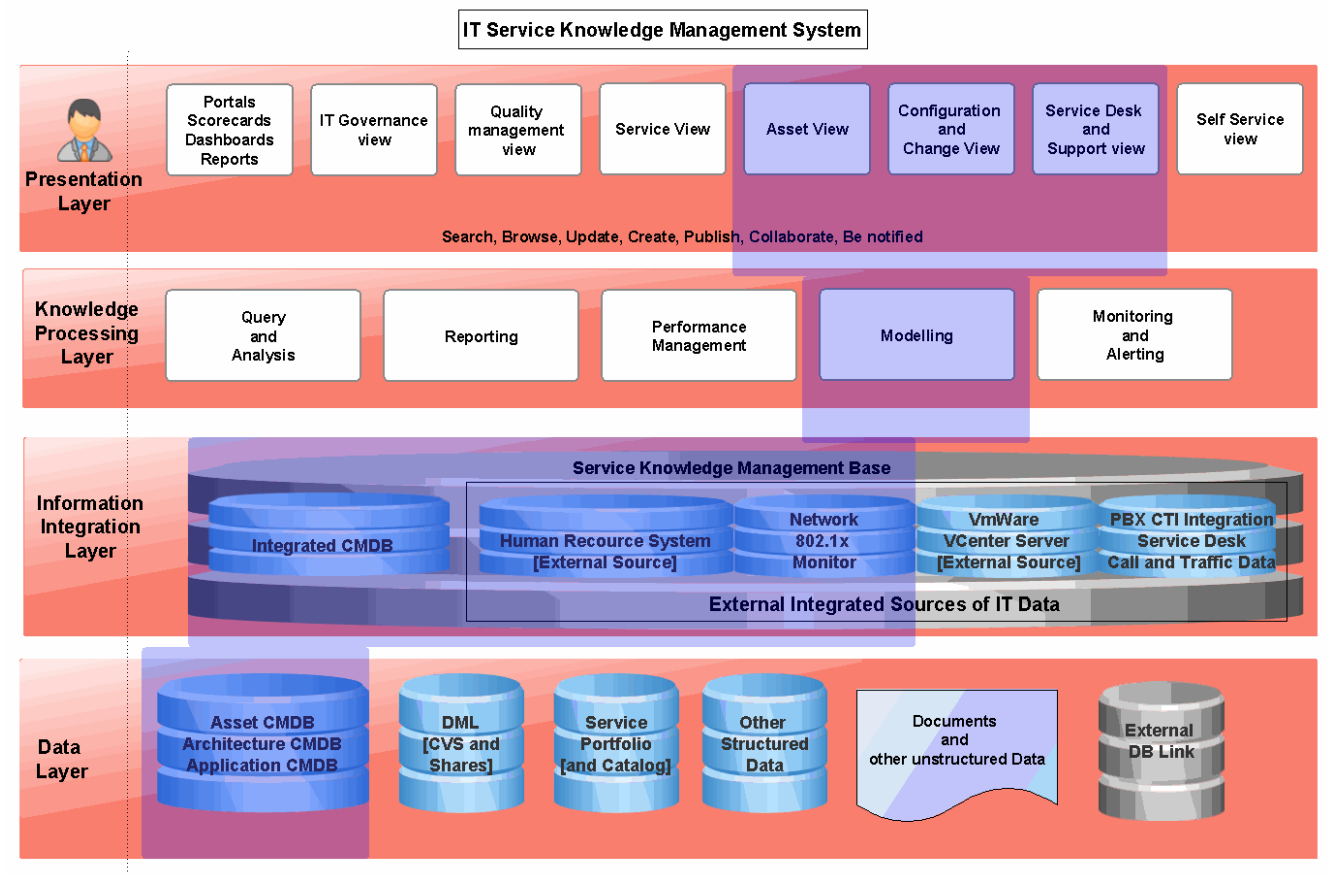


Figura 2 CMDBuild Senato V1.0: First Domains in roadmap

Behind these domains there are a lot of data to manage, the CMDB.

Version 2.0

Version 2.0 is the natural sequel of the project. Once IT operations are covered, the need to measure and control, at the many level of the organizations, grows up. So requests of **reports**, dashboards and so on naturally start. This is a long lasting, even if not so critical phase, which can be related to the **Continual Service Improvement** and **Governance** aspects if ITIL.

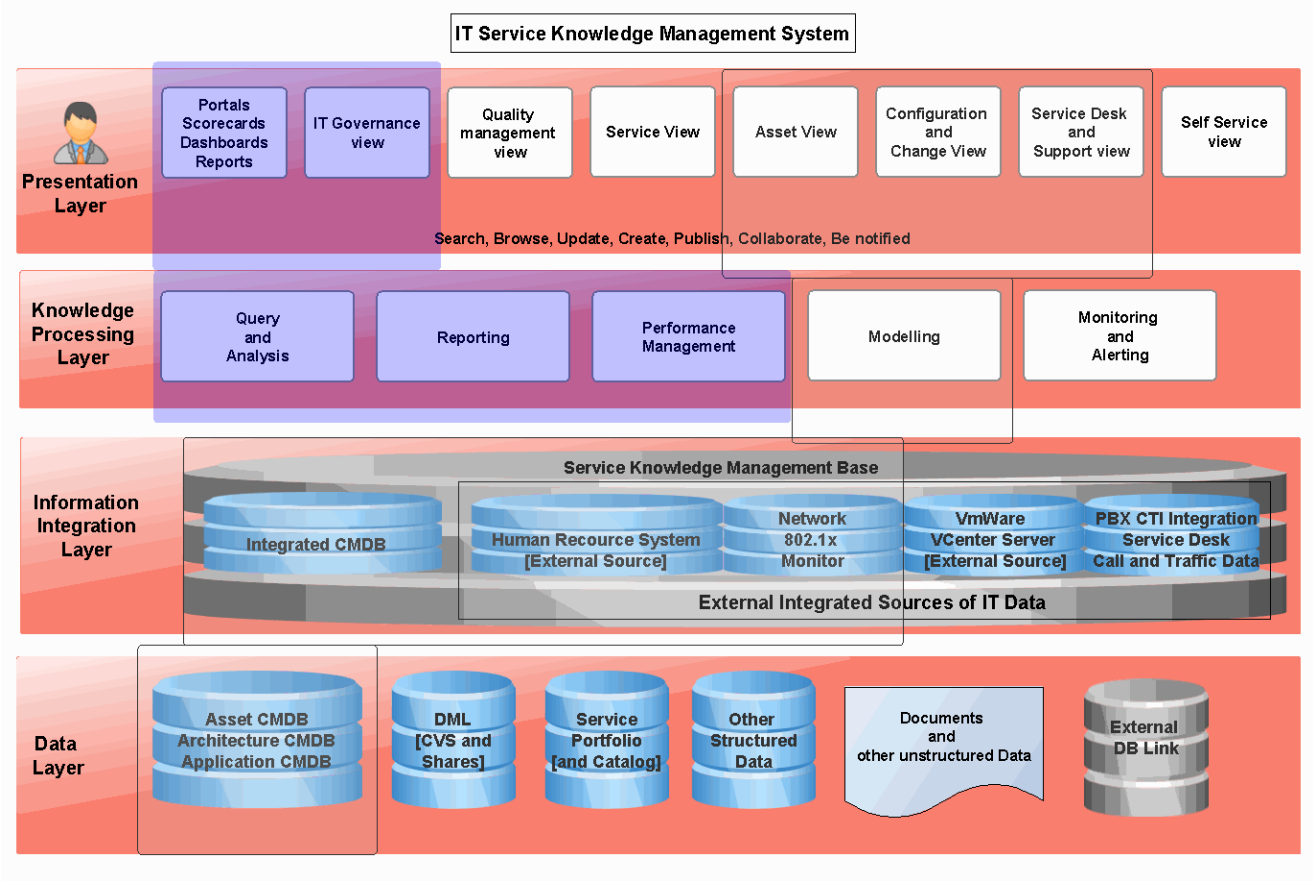


Figura 3 CMDBuild Senato V2.0: Second Domain (darker region) in roadmap

From this phase, a tuning and maintenance activity starts, to better adapt and tune the tool to user needs.

Version 3.0

Version 3.0 is the another **big organizational change.** It's quite complex as many pieces, also from the previous phases, must be in place to work correctly. This phase covers the **Service Portfolio/Catalog** and everything goes around it.

An added value for the end users is the availability of the **self service portal**, which could be the small IT cockpit the IT department gives to them (a new service in itself).

The introduction of the service portfolio and service catalog means a lot of internal work to give structure to the "Service", find the right internal definition of a service, discover and list all the services, link them to applications (another CI type to list) and other CIs, and define the visibility rules for the end users on the self service portal.

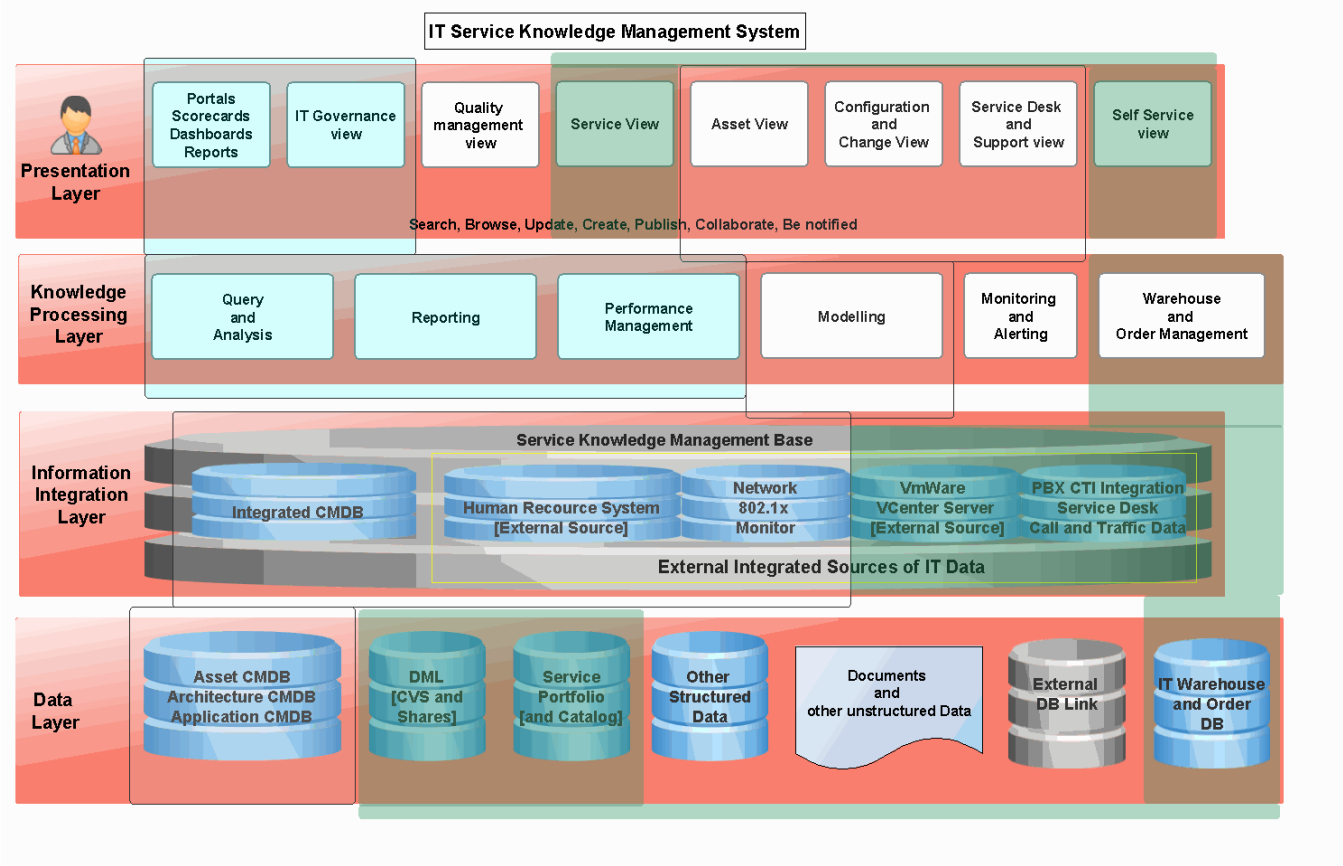


Figura 4 CMDBuild Senato V3.0: Third Domain (darker green region) in roadmap

Walking on the road(map)

The various versions could not follow a strict sequential timeline, as low priority parts of a previous version can be developed in parallel with parts of the new version.

Other important domains, which are not evident from this diagram, are normally developed on the go, along with the main versions. These domains are related to Change and Configuration Management, Release Management, Access Management and Monitoring and Alerting.

Another quite big aspect is the integration of the SKMS with external IT systems. In an IT provider, there are many systems that can produce data that could or should be inserted automatically in the SKMS. A specific chapter is devoted to this theme, which can be considered another subproject (or even more than one) on the project, for the effort needed.

7 Project Management Approach

An ITIL SKMS and CMDB project is a complex endeavor, with many difficulties on all levels and a strong organizational impact. All this complexity must be managed through the right project management approach.

As usual there are the following subphases on a project like this (for every iteration version in roadmap, or even part of it):

1. **Analysis**
   - **1.1. Of concepts**
   - **1.2. Of organization**
   - **1.3. Of processes**
   - **1.4. Of Status quo in terms of tools**
2. **Data Transfer**
3. **Process Implementation**
4. **Test**
5. **Transition**
6. **Production**
7. **Maintenance**

Every subphase can be insourced or oursourced in many different ways. However some lessons on the field have been learned that proved to be useful.

Critical Success Factors

1 Commitment from Top Management

The **first (most) critical success factor** is, as usual, **strong and durable support to the project from top management**. When the other stakeholders will take too much time to respond, do not cooperate or do not make what it's expected to do, the senior manager must kick in.

2 Champion Users

The **second most important critical success factor** you need, are **the champion users**, the ones who will help you during all the project, because they have the major stake. They want the new system working asap, because it will really solve their problems. **These people will be invaluable** in all the phases of the project, practically (un)official members of the project team. These people are even more important in this kind of project, where stakeholders are all IT people. Let's understand why.

As IT competent, IT people suppose to be competent on everything about IT, so they'll try to impose their solutions on the project (because they are supposed to be competent). Really, they are among

the worst stakeholders you'll have to cope with and no, they are not competent on this project, because they are users and this is not their project or their job. Champion users are your best allies in these situations, acting with their diplomacy, in the field of their competencies (as users) and on the other colleagues.

3 Quality of analysis

The **third critical success factor** is **the quality of the analysis**. The analysis phase is extremely critical and time consuming. The results of the analysis, the overall requirements, must be very precise, otherwise the implementation phase will put too much creativity to fill the gap, and the result will be a mess in the best of the hypothesis.

4 Sound Concept Schema

The **fourth critical success factor** is **the overall concept schema**. All the relevant concepts must be identified with all the relations. This implies the necessity to make a lot of interviews, with all the IT people, to produce the big schema. This schema is the foundation of the CMDB and of all the other information necessary to the execution of processes and IT operations.

Then, once a domain in the roadmap is selected, the concept schema can and must be enriched with the many attributes, which will naturally bring you towards the logical DB schema.

This CSF address point 1.1, **Analysis Of concepts,** of the project phases. It will take some time, but not a lot. You need to make it first, because this concept schema will be the foundation of all the rest of the work, along all the roadmap. It is the nucleus around which all the growing system will run, and must be sound and rock solid. However It's not necessary to go deep on every part of the conceptual schema at the same time. What is important, in this phase, is to have the overall skeleton schema to cover the roadmap. Detailed specifications will be added domain by domain, on the road.

If you have a very good (and costly) supplier, you can outsource this step, otherwise, having internal capability, it is far far better to insource it. Internal people can drive and understand better and quicker internal needs. The risk of delegating this work to outside people is the long time (and cost) it can take, to really understand the specificity of the IT provider. Worse, the supplier could impose its vision of the CMDB concepts, which doesn't necessarily adapt to your reality and eventually will be rejected by the organization.

Processes and Organization

The next two steps in project phases, 1.2 **Analysis Of organization** and 1.3 **Analysis Of processes**, are about **the analysis of processes and organization**. This is an **extremely delicate CSF**.

Here the real interaction on sensitive arguments start within all the IT organization.

The formalization of non always formalized processes and organizational groups arises, with some surprises and a bunch of different visions on how things are or should get done, by the many stakeholders.

The unofficial micro organizational structure becomes evident, stepping roughly through processes, and at the end a list of "who does what" is elicited. The result of this "first pass" is a sort of RACI

matrix, or many of them (one for every process in the analysis domain), from which a rough list of organizational units and activities can be produced.

Detail, detail, detail

Now you have **the list of the real OU**, their generic assignments and you have to start go deep in every process. This is the final objective of these two steps, before proceeding with the implementation.

This is really a very difficult part, because now the process analysis must be very detailed. The level of detail must be good enough, to support the implementation on the platform. **The level of detail must bequite high**.

User validation

This **analysis must be validated by the users** too.

These steps are very time consuming, mostly because there are many different users, which have many different and strong views on how the same thing gets done (or should be done). But they have no time to deepen with you the nitty-gritty details of the analysis. Nonetheless **you need their validation** before starting the implementation.

The (anti)pattern is: the supervisor explain his point of view about what the reality should be (vaguely). You later discover that things aren't exactly that way, but he doesn't know. Now the doubt is: what to put in the analysis, the real, the story or whichever?

Everything get solved very slowly (remember, they have no time for you):

- with the help of the champion user, who trades with the supervisor the process structure and let you get a firm on the analysis
- with the tracing of requirement changes, even in late development, even after the firm on the analysis.

Insource or outsource this phase? If you have internal competences, no way, insource it! Outsourcing to an external supplier all this reworking could cost a lot of money and the results could be a waste of money, as the user will never be satisfied.

Existing tools

Point 1.4, **Analysis Of Status quo in terms of tools,** of project phases is **another CSF**, to verify if there are **tools**, which already support (even partly) some processes in the IT organization, which must be phased off or coped with.

In case of **full replacement of a previous tool,** two **problems** must be solved:

- **All the functions of the old tool must be present on the new,** with some gain to justify the phase off
- **All the relevant data managed by the old tool must be migrated** in your brand new CMDB schema supported by the new tool.

Data Migration

Point 2, **Data migration,** which applies if there is a preexisting tool, **is another CSF,** as it is a very difficult task. It can be considered a project in the project, a subteam of the main project team must be dedicated to it. This dedicated team will have to:

1. Understand the (source) tool data structure and the (destination) new data structure
2. Identify with the user the data to be migrated, normally CI
3. Understand the source data and their quality, often not at its best level
4. Prepare mapping tables of data from source to target, both for data structure and data values
5. As the source system uses internal unreadable Id as key for records, the same normally happens in the target system, identify business key fields on data, to replicate relationships and integrity constraints
6. Define the sequence of concept migration as a directed acyclic graph, so every concept is migrated after all its parents concepts are already in place in the new schema, to replicate 1:N relations
7. Verify which cyclic or N:M relations must be updated after the migration of all the concepts
8. Prepare ETL and temporary migration structures
9. Make tests and modify until ok

Point 3, **Process Implementation**, and point 4, **Test**, are the core of this booklet (see next chapters).

Point 5, **Transition,** will not be addressed in this booklet as it is another big e very critical subject (probably next version of booklet).

Point 6 and 7 are ordinary activities.

Project Team Structure

A **project manager** is obviously necessary and must be high level in the provider organization with a strong technical involvement.

As an alternative, this role can be splitted between two people, where the high level project manager is supported by a very competent business process analyst, who know well the target platform too. The two must behave like a single logical unit.

The **data (sub)team** is dedicated to the data migration subproject, if necessary. It can be composed of a single person, if very competent (it depends from the effort planned to migrate too).

The **development team** masters the production of workflows and the customization of the platform. The suggestion is to insource or cosource. Full outsourcing can be risky, as you risk to lose visibility on the tracing of requirements from the analysis. The management of the quality of artifacts could also be difficult to impossible.

The number of members of the development team depends on the number of processes to be developed in the round and the complexity of the concept schema. A competent team leader is necessary to supervise the work of the group.

The **champion users** are also members of the project team (sometimes de-facto), even if they don't respond to the project manager. They cooperate intensely with all the rest of the team and, in fact, they are one of the best resource of the project.

Some ITIL Project Management Advices

During this project, experience confirms the goodness of the many good practices about software development and organizational change management.

Among the others, in this experience, some learned lessons reveal their importance:

- **Trace everything**, specifically requirements, changes on requirements and resulting consequences on development times
- **Make periodic meeting** with the team and users
- **Trace and write** interactions on decision with the user.
- **Pretend user acceptance** on every phase, steps, analysis artifact of the project
- **Use email to trace, phone call to explain**
- If some outsourcing is present, **the user should never contact directly the outsourcer**, as the project team could lose track of requests and bugs.

To summarize, simply stated, one principle can follow from these advices:

Trace everything important and keep control of communications.

8 Project of the CMDB

Conceptual schema

The conceptual schema of the CMDB is the cornerstone of an ITIL automation system.

In the schema all the relevant concepts must be defined, as on this schema all the ITIL processes will operate.

2 Steps to a Schema

The schema analysis cam be operated in **a couple of steps**, that can be iteratively repeated for every domain in which the roadmap is partitioned:

1. **First pass, overall analysis with the definition of the overall concepts**
2. **Second pass, refinement of details: attributes, grouping e more specific relations**

Overall analysis

The overall analysis phase of the schema is critical.

The main artifact of this phase is the global schema of the concepts, above all, the Configuration Items which are relevant to the provider. It is not necessary in this phase to go deep in the definition of every attribute or every relation among concepts. **It's extremely important to indentify all the relevant concepts**, even those which will not be part of the implementation domain in roadmap. This schema gives the overall picture of all the CIs, that a provider must manage to deliver Services. It is a map that the on-going ITIL organizational system will have to cover with processes.

An approach that worked for us is this: **the CMDB concepts can be grouped in supersets like the following**:

- The properly speaking **CMDB**, made of **IT CIs**;
- **User and organizational concepts**, that is the model of the organization and all the users
- **Location concepts**, that is all the concepts that pertain to the localization of the CIs
- **Administrative concepts**, that is all the concepts related to the contest of contracts, orders, suppliers, maintenance, asset lifecyle
- **Process concepts**, that is all the concepts related to the executions of management and operation processes

Most of the concepts are CIs, which are components of services (e.g. datacenter infrastructure, network, PC and other devices from the physical world, and their virtual counterparts). They are the target of every other concept in the ITIL Service Knowledge Management System.

Refinement of details

Once the main concepts and their relations are clear, it is possible to develop a more detailed view of all the involved attributes and further detail relations.

Attributes, Attribute Value Domains, Attribute Groups

The CIs (IT CIs) have many attributes, some (most) of which are common to different types of CI.

Attributes can be grouped in many ways, to manage them as clusters, to assign to the CI types.

Two main groups of attributes can be defined:

1. **Attributes common to many types of CI (possibly all)**
2. **Attributes specific of a CI type.**

In the group of **common attributes**, other **subgroups** can be identified (e.g.), as from our analysis:

- **Network**, with all the attributes that pertain to network management
- **Administration**, with all about global lifecycle of physical CIs in the organization
- **Operation**, with all about IMAC processes or day to day management of CIs
- **Warehouse Logistic**, with all about the data needed by the activities of the internal warehouse and external CI maintenance
- **Other**, other attributes both specific for the CI or general but outside the previous subgroups

Every subgroup is normally used by a subset of processes or even by a specific organizational unit.

There are many attributes that can have a value from a predefined domain of values. These domains must be defined too.

Characteristics of DB, CMDB and DBMS (inheritance)

In ITIL, the structure of the SKMS is not clearly specified, even though a notion of federation of many CMDBs is present.

There are many tools offering the CMDB, every one with a different philosophy and sometimes focused on some types of CIs more than others.

From the experience of the writer, there are some considerations that must be made on the choice of the tool and of the CMDB in general.

One-to-One CMDB-DB

An integrated CMDB Conceptual Schema is of paramount importance for every decision. This schema is the reference for every choice, both technical and organizational. This schema gives the high level view of the ITIL system in the provider, and gives indications about the coverage area of both ITIL implementation (Organization) and tool support (Automation).

Even if the IT business domain is more or less similar across different providers, there can be major differences in the way concepts and processes are organized inside providers, so a universal, externally defined CMDB can be useful, but could also be an overkill or could not adapt well to the provider vision.

In a small medium organization, the Configuration Item types are normally less than those you can find in a big company, but enough to justify the adoption of a CMDB. Even their definitions and relationships can be originated and customized from the internal ICT architecture.

Tailor on needs

The CMDB should be a dress tailored on the needs. Needs are different from the big third party service provider, which could adopt an externally defined and "universal" CMDB and, e.g., the very small captive internal service provider (who can work well even with an excel file).

A small medium organization stays in the middle, so, in my experience, the CMDB schema should be the result of an internal analysis where only these relevant CI types are inserted. These types can change in time, as the adopted technology evolves. This schema normally is not so big to deserve esoteric technical solutions.

The adoption of a federation of many CMDB, every one focused on a portion of the infrastructure, is a source of technical challenge.

Keep the focus and evolve

The advice is **to stay with a single DB system** and **make it evolve over time**, whenever possible.

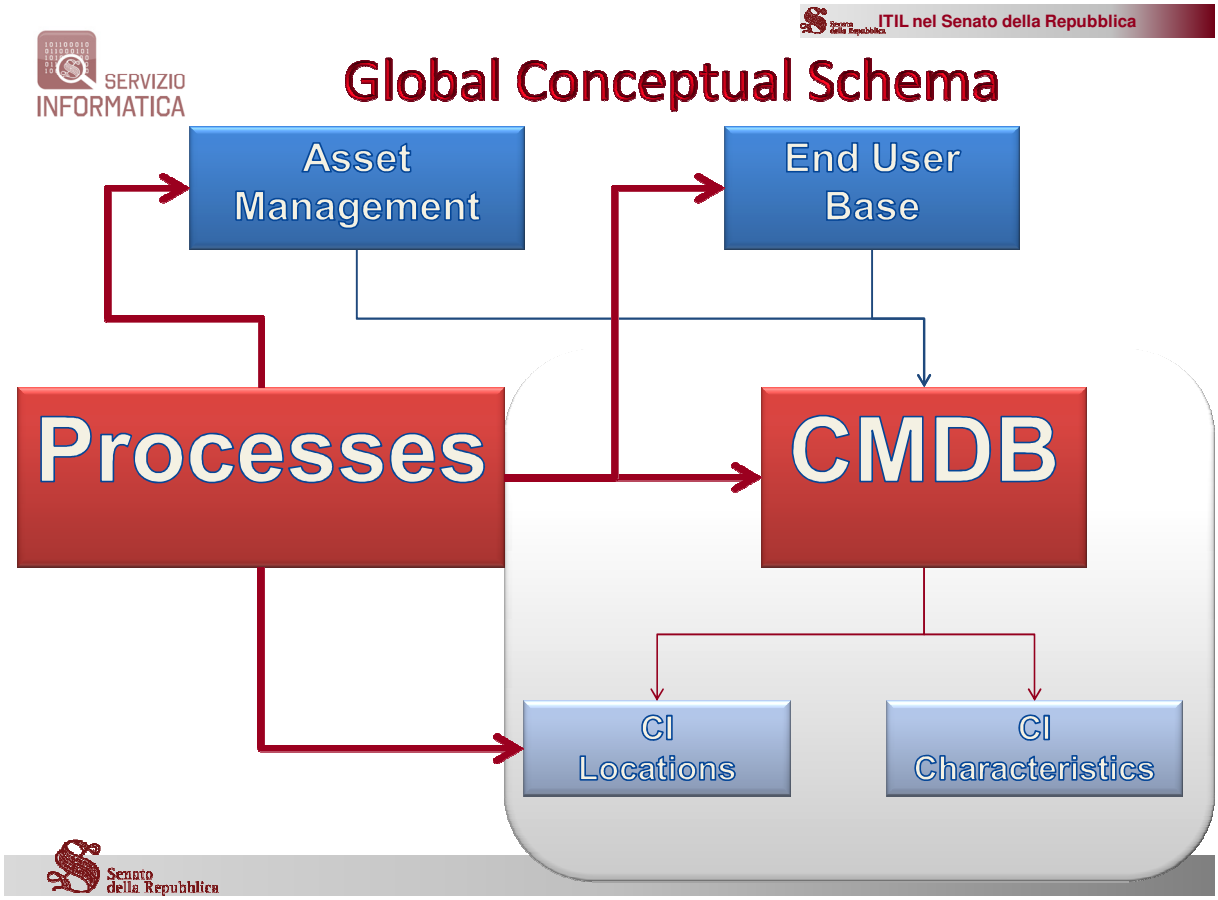
The "big ITIL certified product" solution can be overwhelming, costly and perceived as too complex to give real benefit.

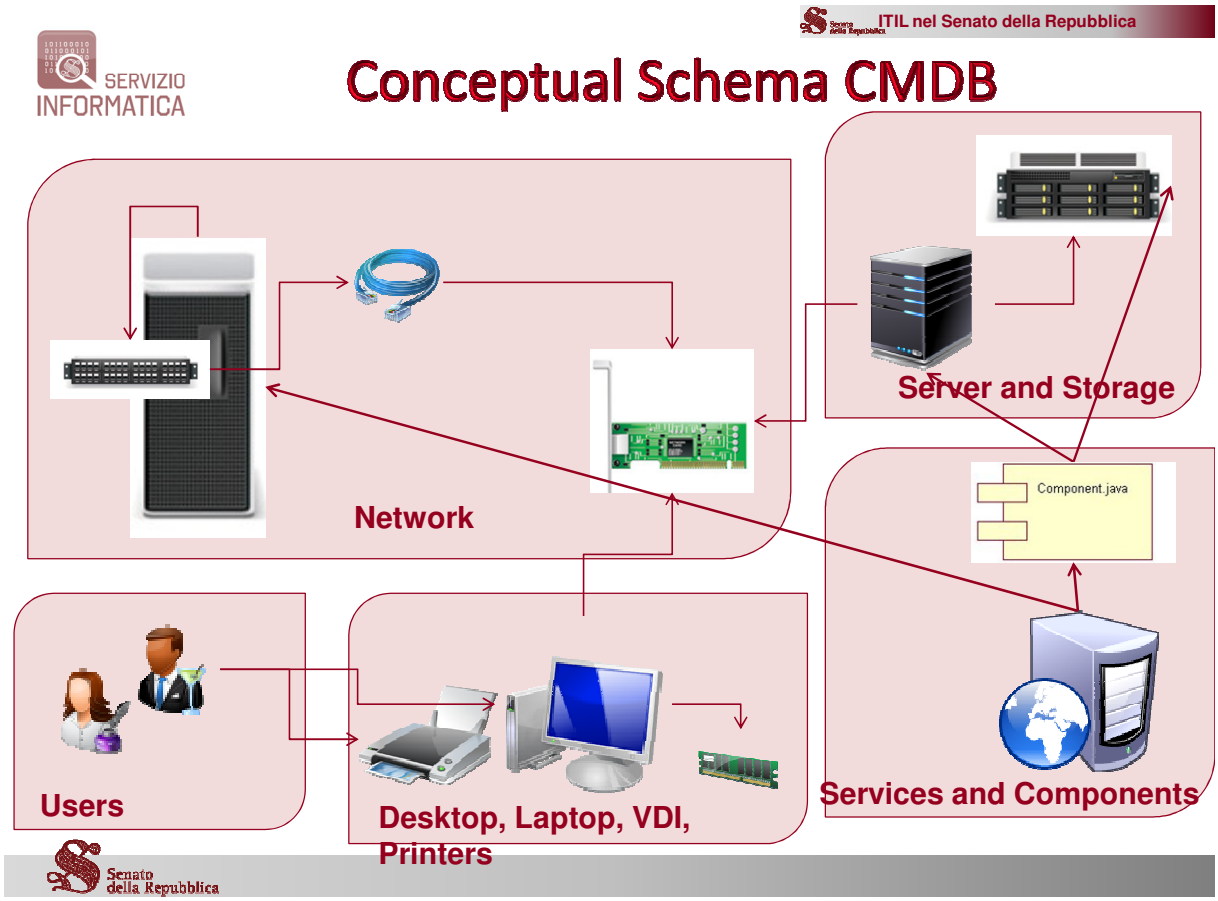
The guideline is "start simple and then add just as much needed", but only after an overall analysis to mantain the project on the roadmap.

Keep the focus. Constantly verify ITIL coverage of the roadmap and track user needs and priority. Change the roadmap, on a provider need basis.

CMDBuild is a good fit to this kind of philosophy.

Example CMDB schema: The Senato CMDB Schema




ITIL nel Senato della Repubblica
SERVIZIO INFORMATICA
Conceptual Schema CMDB
Network
Server and Storage
Component.java
Users
Desktop, Laptop, VDI, Printers
Services and Components
Senato della Repubblica

9 CMDB Issues on data migration and loading from the dismissing system

In an ITIL project the following things can happen (even all togheter):

- You start from scratch, information on CIs are scattered around in paper and excel
- there already are tools to manage some aspects covered by ITIL, which are to be phased off by a better choice
- these tools have a DB of important data, to extract and insert in the new tool/data structure

If you can simply start from scratch and do not have to migrate other tools data (only excel, perhaps), you are very lucky, but normally reality is more complex than this.

Normally, some kind of tool, with a complex DB, is already adopted and must be coped with.

This is one of the most complex, but essential, task of the project and it is time-consuming (it can be a significant percentage of the overall project effort).

Data migration Data Nightmare

There are **many complex issues,** which stratify on the same task of data migration:

- **Data structures are different (both in the metaschema and the platform)**
- **It's very difficult to migrate the relationships**
- **Data domains will be different**
- **Other data transformations will be requested by the users**
- **Quality of source data is normally not constant to low**
- **Some data will not be migrated from the source tool, but from other enterprise systems and then integrated.**
- **Bug in the mapping and ETL procedures**

The most important data which must be transferred from an old to a new system are the CI data. These data are both those specific of the CIs and those of the relations among them, relations with users, with locations, with contracts and so on.

Historical process data are normally not so useful. To simplify, they can be left on the old system, just for consultation.

The **high level migration order** is multistep:

1. **Transfer first the simplest lists (like Model, Brand, Organizational Unit ...)**
2. **Then the referenced registries (User records, location records, ...)**
3. **Then the CIs, in an particular order given by the relations, among CI types, which must be copied.**

In Step 3, the loading order, of CI types from the source schema, must follow a sort of directed acyclic graph navigational structure, to be sure that CIs referenced by other CIs are loaded first. E.g. a network switch must be loaded before the network interfaces it has, because these interfaces, as children of the switch, must reference an already loaded switch, to establish the correct relations.

The **primary keys to CIs** must be identified in the **old and new system**, and a **mapping** among old keys and new keys must be defined and copied in the new system, to allow the rebuilding of the relationships among CIs, in the new data structure. Normally it's not possible to directly import internal identifiers used in the source data structure, for a number of reasons.

Data domains of attributes between systems can be different, due to different data structures or different requirement analysis and implementations in the destination system. One or more **mapping tables,** of attribute values, must be defined from source to destination systems.

Data quality issues

Data quality of source system is never at its best, the migration can highlight this problem.

A good approach to follow, in this case, is to "clean" the data on the source system as much possible, to be sure that the migration procedure will not need to cope with meaningless tuples, so ETL code will be focused only on migration algorithms. The most complex is the ETL procedure, the worst the data verification and debugging issues.

Data migration procedure

The **inner working of the ETL migration procedure** can be designed as follows:

1. Make a set of migration structures in the destination system whose schema is the same of the destination schema
2. Design a procedure for every type of CI, defined in the destination schema, which read from the source schema only these specific CIs, and put them in the migration destination structure. Put in this structure also the data necessary to recreate the relationships.
3. Once all these intermediate structures are filled with the CIs, follow the order of loading of the CI types, to populate the destination schema with CI data and the relations among CIs.

Test Procedure on data migration

The **verification of data consistency** on the migrated data is a critical and not easy task, as source data are far from perfect, but you must be sure that the data in the source and target system are consistent between each other.

The main objective is to have the same information (as is, with is good or bad quality) in the source system inserted in the new one (the same holds if only a subset of source information has been selected for migration).

In our project, one table map drove the migration: **the mapping among old and new types of CI**.

Another table map was used to map the **state fields** of every CI.

Many other complex mappings can be present, everyone must be defined in advance and constantly documented, as it can change during the testing of ETL procedures.

To check that every type of CI has been correctly inserted in its destination class and has the correct state values (and correct values of all other field transformations), the correct relationship and the correct field length, **a set of pilot CIs,** from every class and with every relevant field combination, has been identified.

At every trial of data migration, the pilot CIs in the target system were checked against the pilot CI set, written on a separate Excel sheet, for perfect matching or (sometimes) errors.

10 Activity Domain Approach

Requirement Analysis

The implementation domain is quite large and has been addressed by dividing it in many sub domains, or **domains of activity**, as seen in the previous chapter.

A domain of activity (Activity Domain) is composed by all the concepts and processes that combine together to make a certain job done.

E.g. Ticketing, in the sense of Request Fulfillment and Incident Management, is an Activity Domain that makes sense to address in a single analysis and implementation iteration.

From the roadmap, a set of Activity Domains are determined and prioritized.

Every domain is singularly approached to elicit the list of generic functional requirements, the conceptual schema and the set of processes acting on the schema concepts.

A **context analysis** is also performed to determine which **other domains cooperate with the one and in which way**, to evaluate the reciprocal side effects, to account on during the development of the one. These side effects can generate later changes in analysis and implementation, so it's useful at least to have an idea about them.

A deeper neighborhood analysis is useful mostly on the already implemented domains. Having an upfront idea of side effects on already deployed domains, helps smoothing the integration of the new domains, in a consolidating configuration.

Package Concept

The **Activity Domain** modular approach produces a set of artifacts, that are considered a **Package**, like a SDP (Service Design Package), to be deployed and transitioned on the automation platform.

The **artifacts** in this **Package** normally are (not necessarily all of them):

- **A list of requirements**
- **A conceptual schema**
- **An organizational analysis**
- **A set of cooperating processes, documented trough standard documents**
- **A database structure and configuration data**
- **Scripts and procedures in the languages of the platform**
- **The implementation of processes, with all the deliverable managed by them**
- **Some reports**

The Package is the basis of detailed planning, supplier management, configuration management and all the other process in the ITIL design and transition service phases.

The Package is traced (as much as possible, at least) from the requirement to the production phase, as a modular architectural and organizational unit.

11 Activity Domain Organizational Groups Identification

Adopting a process approach needs a careful analysis of operational groups involved in the process activities.

These groups normally are more fine grained, respect to the official organizational schema.

These groups could already be described in one of the following sources:

- in preexisting automation systems,
- in the authorization role descriptions (the set of privileges of users of the systems),
- mentioned in other formal or informal sources (like IT internal documents, wikis, mail groups ...),
- unofficially adopted by the organization to have things done.

When a process analysis starts in an organization, addressing the adoption of a workflow automation tool, **all these groups must be identified**, univocally named and given a sort of official "status" in the organization.

This is a very difficult phase of the analysis, both to gather the information and for the **"political" implications**. The real workflows will come to light, with some surprise too.

Operators and users normally will be part of many groups/roles, as they execute many activities in different situations, even on a calendar basis. They can take part with different hats in more than one process.

The number of these groups could be significant. ITIL defines a lot of them, however every provider has its characteristics.

What must be documented and clearly defined, it's the correct identification of every group, its assignments in every process and the information/CIs they treat.

Organizational Change Management

This paragraph is just a note about an important activity, Organization Change Management, which pertains the ITIL Transition phase. It's a vast subject and will be treated with the Transition.

12 Activity Domain Process Analysis

Platform drives

The methodological approach to the ITIL SKMS implementation is designed and driven also by the characteristics of the automation platform.

The level of customization of the platform and its features should be well known in advance to be sure that the requirement analysis could be implemented by the platform.

Even the template artifacts, created to guarantee a uniform structure to the analysis content, should be designed according the customization points of the platform.

I understand that this approach can be considered less orthodox by conceptually oriented analysts. In my experience, this pragmatic approach guarantees the best quality of results both to users needs and to platform features.

This is a different approach to a classic process engineering (or reenginering). Process (re)enginnering, as a discipline on its own, is performed without considering the target platform (if any) that will support the automation of the workflows. This approach, even if methodologically sound from a theoretical point of view, shows its limit when an automation platform kicks in, in my experiences of projects like this.

The implementation phase needs a lot of specific and precise information, not always present in a purely conceptual analysis if not targeted to a specific platform.

The organizational designer should instead know to the best level which information, which detail, which structure is needed, to produce a quality implementation on the target platform from his analysis, and orients his analysis accordingly. This approach helps to speed time, too.

If, e.g., the platform couldn't print a module of user intervention on site, even if we could argue that the platform choice should be reevaluated, it would be a waste of time designing the best module ever, which would remain useless on the analysis document.

So, again, the structure of the analysis (that is the set of information to gather and the way to collect it in an organic artifact) should be designed having in mind the target platform and its features, to shape the structure and the detail of the conceptual information accordingly to them.

This way the analyst can be sure to produce the most complete information, needed to proceed smoothly with the implementation, to take advantage of these features or limit some user requirements, which can't be supported by the platform.

Workflow Analysis and Design

The core of the automation system are the workflows which describe the processes.

Normally the platform has a designer tool to manage the workflows as diagrams.

Using the platform designer to make also the workflow analysis, make it faster and coherent with the capabilities offered by the target platform.

In my experience, it is useful to adopt this same tool both for the analysis and the development phase.

This approach has both advantages and disadvantages.

The **pro** of this approach are:

- the conceptual workflow very probably will be adapted easily to the development phase on the platform.
- the workflow is already in a format apt to development, that is you don't need to retype the diagram in a different format just to make it available in the development tool, as it is the same used in the analysis.

The **cons** are:

- The conceptual level of analysis is biased by the platform, and for a purist of RACI matrix this could not be acceptable.
- The documentation of requirements of these designers can be poor, so you need to adopt an external support to describe the detailed process requirements to complement the diagram.
- These designers, as development tools, don't always have a process repository to store workflows, but work on separate files.
- Tools, aimed more at the analysis phase, populate their repository of organizational processes, where every aspect of the analysis is inserted, can be referenced and can be used in other processes, to have a global and coherent vision of all the organizational units, the processes and the activities they are involved in.

In the case of IT processes, a "lean and mean" approach has been preferred, to limit the variety of analysis artifacts and try to go faster, considering that the analysis phase is critical for the success of the system and can be complex and so long.

Documents managed by a process

In a process there are **2 main types of information** to manage:

- **Structured data**
- **Documents.**

Structured data are process information normally stored in a DB in tables and fields (see specific chapter).

Documents are unstructured data that is useful to access in the process or are generated by the process itself.

Pdf or Word documents can be **an input to the process** for later consultation.

The process can produce also **output document**, like report, worksheet to be signed by the end user and then scanned and inserted again in the process.

These process generated documents (the notorious "reports") must be precisely defined, to verify that all the necessary information is already tracked in the process or in the CMDB. Their templates must be designed too (end user loves making this kind of creative activity).

Process Analysis Template

From the features of the platform and the needs of the requirement analysis, a Process Analysis document structure has emerged.

This structure is versioned as it is constantly revised and adapted to changing needs or to new platform features, that can be important to account during the analysis. The consequences of new features, in the analysis phase, can be the opportunity to adopt the new functions and/or parameters, which must be defined during the process analysis.

In practice, it's completely useless to foresee the management of lists of CIs in the process, in a way so superb and esoteric that it isn't supported by the platform. It's a waste of time. The platform can manage lists in a number of ways, the best of which depends on the requirements.

E.g. Now Cmdbuild has a fantastic grid which can export CI data in excel format, let you modify them adding column values in batch, merge back the new file with the data already present in the grid. This is a feature very useful in order management of many item of the same kind, where the supplier gives, in file format, the list of item specific data: serial number, part number, mac address, ... This is a feature which drives the analysis in a very specific way: the template can keep trace of this too.

In the definition of the Process Analysis Template, keep always in mind the platform (be pragmatic), but make it in a way which guarantees that the analysis is as complete as possible (be conceptual).

The Process Template Document is described in the appendix.

13 Activity Domain: Identification of Process Concepts

Concepts managed by every process

As we already specified before approaching processes, two aspects should have been roughly sketched at the start of the project:

- **The CMDB overall schema**
- **The set of Activity Domains in list.**

The detailed set of concepts, managed in a domain, must be identified during the analysis of the domain itself. These concepts must be arranged in a conceptual schema.

The set of processes in a domain must be identified and then checked against the domain concepts, to assure the consistency and completeness of the domain in terms of processes and their concepts.

Every process, in the domain, will act on a subset of the domain concepts. This subset must be identified to be sure that every process has access to all the data it needs.

At the same time, all the process of the domain should cooperate on the same set of concepts/data, to maintain the coherence of the Activity Domain.

Every process will have also its set of specific attributes/data, necessary to its execution.

CMDB harmonization and integration of concepts

The concepts managed by domain processes can be already present in the CMDB.

They could need some modification (normally addition).

They could be brand new too.

The Activity Domain concept schema must be matched on the global CMDB schema, to identify the gaps and the consequent changes.

Applying this integrated approach, the global CMDB schema coherently adapts to cover the requirements of the Activity Domain processes.

14 KPI and Reports

Service Desk KPI

Once processes are up and running in the system, soon the need of reports and measures arises.

Specifically, the Service Desk, with the management of incidents and requests, is a main source of operational data.

The Service Desk is outsourced, so some reports with relevant KPI are both useful for operative reasons and contractual verification.

Operational KPI under control are:

- Overall Number of requests per month/year
- Number of requests, for every classification type, per month/year
- Number of request solved by every Service Desk operator, per week/month/year
- The distribution of requests on every hour/day of week, to analyse user support traffic and workload

The same set of measures/reports is present for the incidents and for the aggregate of the two, incidents+requests.

Assets Reports and KPI

There is also the need to keep under control assets, that is, physical CIs assigned to users.

This aspect is important as we have defined plafond for groups of users. Certain groups of users can have a predefined maximal number of IT resources (e.g. PCs or notebooks), independently from the effective number of users belonging to the group.

When a request of a new resource arrives from one of these groups, for example, a check must be made to make a go-no go decision on the new assignment.

The number of assets assigned to every group is an important KPI, to control on the limit forced by the plafond.

As some groups can be also dynamic, so new users arrive and old ones go away, another important periodic control is the trace of asset assignments to users, to be sure that an asset is assigned to an effective user, or perhaps it must be retired or reassigned to the correct person.

15 External and Business System Integration

External systems

Even an ITIL automation tool is an information system inside a provider that needs integrations with preexisting systems, to operate more efficiently.

The first systems to integrate are:

- The Authentication system, in our case the MS Active Directory Domain
- The email system, for CMDBuild notifications
- The Business User Record System (or the HR system), to update automatically the user base.

Many other candidates are in roadmap:

- All the infrastructure management systems (e.g. Vmware, SCCM, ...), to gather automatically data for the CMDB
- Monitoring systems, to open automatically tickets in case of major failures
- Supply chain management system (for orders and suppliers)
- Inventory system (for asset lifecycle)
- PBX, to improve the efficiency of the Service Desk
- Facility management, for location updates.

Business User Record System (HR)

A connector with the User Record System has been implemented, to synchronize the user base created in the platform with the company employee record system.

This connector synchronizes both the organizational structure, at first, which is partially replicated in the platform, and then a filtered list of users who are served by the provider.

The data replication policy is the following:

- Every change of new or modified organizational groups in the company record system is replicated in the platform (in a simpler structure, good enough for the IT process needs); deletions are not replicated.
- Every change in personnel, consisting of new arrivals or changed positions, is same way replicated in the platform; deletions are not replicated.

- Changes made in the platform are not reverse replicated and can be overwritten by the replication process.

So the replication process is one-way from the authoritative company source. The deletion are not replicated for two reasons:

1. Their replica is more difficult to implement
2. Deletions can give rise to problems due to existing configuration items still related to deleted positions (e.g. a notebook still in charge of a "no more" user, who must give it back).

Email

The platform is integrated with the email in many ways.

It can send notification email to end users, to keep them informed of the status of requests.

It can send notification email to selected IT groups, to signal the need to take care of an activity or simply to notify an event.

It can interactively manage an email thread, during an activity to ask or give explanations to the involved end user (in experiment).

It can receive formatted email to start processes like opening an incident from an anomaly notification coming from the monitoring system (in experiment).

16 Product Quality

Approach to Product Quality

The approach to quality developed during the project is based on the following practices (not always an easy accomplishment):

- Requirements, features and bug tracing
- Periodic verifications of requirements
- Test plans, Tests and Test results verifications
- Definition and enforcement of operational/development procedures
- Style rules and Code inspections

Periodic verifications are made on deployed processes, to verify that they really implement correctly all the requirements. In case of non-conformities, or improvements, (corrective) actions are arranged.

Verifications are also made on the roadmap, to verify the progress of coverage of functionalities, or to change priorities and replan.

Style rules have a dedicated chapter. Code inspections are useful to verify the respect of style rules and sometimes pinpoint implementation decisions that need some reworking.

Management of development, test and production environment

We have 3 virtualized servers for CMDBuild: one for internal development, one to make tests on our and supplier customizations, and the production.

The Test Server is a copy of production and can be scratched and rebuild in any moment.

The Development server contains on-going new and maintenance developments. Once the package is complete, it is deployed first on test, to verify both the functionalities and the correctness of deployment. This last issue is due to the peculiarities of the platform, CMDBuild, which is DB based. So we developed practices, with the supplier, which enable us to identify the new "delta" of development, create a package and deploy it on another environment, without worrying about the entire DB (which would be not feasible in production too).

Software Configuration Management Procedures

Packages are versioned and saved in a Configuration Management System, to be sure to trace major modifications and have a unique common repository of sources.

We adopt CVS at the moment.

Processes in particular are versioned with care, as they can often have maintenance.

Outsourced processes are very critical about versioning, as there is always the risk to lose control of development in the test phase, where the bug queue can grow and shrink quickly. In this particular situation, we asked the supplier to put a prefix to the name of process related files, to trace exactly the version, so everybody is sure to work always with the newest files.

Production deployment Procedures

We adopted the Package as a deployment unit (as defined in a previous chapter).

When the package is ready, it is firstly deployed in the test environment.

Once tests are ok (both functional and of deployment), the package is promoted to production.

To modify production, in case of a major release of functionalities, particularly if new, the activities are made by the IT production team, with the support of the project development team. The deployment activities are performed in specific time-window, during the workweek, for security reasons.

If there are simple bug resolutions or ordinary administrative tasks to perform, the lean approach is preferred: intervention is made during lunch time of workdays.

17 Style Rules

Why Style Rules

Style rules could be considered a secondary aspect of the project, almost a gold plating.

In reality they are extremely important for a number of reasons.

The team is composed of many people, both external and internal to the organization. Everyone could (and would) adopt his own approach in everything, from analysis to code.

Underestimating the management of this diversity can generate, slowly but constantly, a maintenance mess.

To contrast this tendency, it is of paramount importance to define rules on the production of analysis and implementation artifacts.

Every person of the team should know which is the single, correct way to produce a certain artifact. All the types of artifact should be produced uniformly across the team.

The purpose of the style rules is focused to let every member of the team understand the artifacts he needs, even later in time, and put hands on them coherently too. The advantage of this approach is evident in many situations (e.g. when maintenance starts, or when there is turn over on the team).

Style rules are defined and adapted as new requirements emerge, to improve (or at least maintain) the quality of product.

The "**Style Rule Cop**" is a necessary role, as **Style Rules must be periodically remembered and enforced** and someone must do it.

In the following, there are the main style rules which prove to be useful in the project on the CMDBuild platform.

Code rules in DB

Tables and relations of a package must have a common prefix in their name. This prefix identifies the package and helps to group all the elements of a package created in the DB.

E.g. All the classes and relation about IMAC processes have the "IMAC" prefix in their name.

This way it is easy, even by a simple alphabetical sorting, to have the group of related concepts in the DB, about a certain domain.

Style rules in process diagrams

Organizational Lanes are horizontal.

Lines of transitions are straight, vertical and/or horizontal and have 90 degree vertices.

Conditional transitions must be labeled, so that the condition is readable directly on the diagram.

Every set of conditional transitions must have an "otherwise" transition (with its distinctive color).

Colors are used to identify the kind of activity of task:

- Script Tasks, that is those having code and not interactive, have a specific color (RGB (255,255,204), sort of light yellow)
- Remediation tasks or data verification and correction tasks have a specific color (RGB (255,153,153), sort of red)
- Warehouse transaction tasks (those couple of tasks where a group exchange goods with the warehouse) have a specific colour (RGB (204,204,255), sort of violet)
- Interactive task have a specific colour (RGB (187,247,190), light green)
- Block activity have a specific colour (reddish)

Script activities, which create or update DB data (e.g. CIs), should be concentrated in strategic point of the process (if possible at the beginning and at the end), to ease manual data recovery operations in case of process failure or user abort.

Naming rules in processes

The process is a sequence of activities. The following naming rules have proven to ease tracking of items, from analysis to implementation to fixing, and guarantee an enough uniform production in the team.

Every **process activity** as an **Identifier** made of:

- a **code prefix** with a **capital letter and a number** (normally "A" in interactive/script activities, also "S" in scripts),
- a **point** "."
- a **description** without space

e.g. "A1.MakeThis"

and a corresponding longer Name as user friendly label, similar to the Id, but with spaces around the point and the words e.g. "A1 . Make This".

This Identifier must correspond to that used in the analysis document.

This rules help:

- to **trace the specific activity** from the analysis to the final implementation (in the xpdl file) and
- to **have a unique Identifier** for everything **related to that activity**, like **bugs or enhancements**.

So the **implementation** can and must **follow strictly the structure of the analysis**, in the case of the process.

In **case of big changes in the implementation phase from the analysis**, the tracing must be preserved in the following way:

- **Correct the process analysis document**
- **Follow the naming rules to trace the correspondence among analysis and implementation**

In case of **an activity is composed of many sub activities** (sometimes defined as a block activity) during the implementation, the **sub activities** should have **as prefix, in their name and Identifier**, the **Identifier of the enclosing activity**, to trace easily through the various artifacts.

The **process artifacts** specifically are:

- Process analysis document
- Analysis Diagram
- Implementation Diagram (derived from and traceable back to the Analysis Diagram)
- Interactive forms of the corresponding activity for end user.

Every **interactive activity** lets the user work on a subset of the process information through a subset of attributes.

The order by which these attributes are shown on the activity form is the following:

- **Primary characteristic attributes**
- **New attributes**
- **Other primary attributes**
- **Other attributes**

Primary characteristic attributes

They are always present on every process activities. They are the most important as they characterize the process and help the user identify the kind of process and all the relevant information for that process instance.

New attributes

These are new in the process and normally must be acted upon by the user, are often required too.

Primary attributes

These are always present in all process activities but are considered less important of the characteristic ones. After the first activities, they are often read only.

Other Attributes

All the other not classified.

If a new attribute becomes present in all the following activities from a certain step, it is added to the Primary characteristic list.

The position in the list (and in the form) of the attributes should be always the same, across all the activities of a domain group of processes.

This structure of attributes is to make the user comfortable with the tool, to show always the primary information in the same order and position in the form, followed (vertically) by the new information.

A set of common attributes to activities in many processes are, in the following order on the form:

1. **[Process] Message (if make sense in the activity),**
2. **Process Number (it could be a Ticket number)**
3. **Operator in charge,**
4. **Title of process instance,**
5. **Operational State of process (if make sense in the activity),**
6. **Notes, or comments.**

Obviously the list depends on the specific process or group of processes.

All the logic or flag variable/information, inserted by the user on the form, must be rendered by an explicit lookup with the "Yes/No" choice, or similar.

Recurring Pattern and Best Practices

The analysis is organized through Activity Domains. Processes belonging to the same Activity Domain often have many similarities.

These similarities give origin to recurring patterns in flows, which are useful to identify, to implement the same requirements in the same way across all the processes of an Activity Domain.

The same apply to scripts.

The workflow tool, adopted by CMDBuild, can include libraries of flows to ease this modular approach to process flow development, but CMDBuild doesn't support it at the moment.

The suggestion to accomplish reuse, for the **subflows** and the **scripts**, is specific to CMDBuild, given this limitation.

The **subflows** can be reused through a careful cut&paste on the xpdl source, from a set of code snippets you can collect.

The reuse of **scripts** can be obtained by inserting the beanshell source code, to be reused, in DB text fields of tables devoted to this purpose, loaded and evaluated by the beanshell at runtime, inside the process script activities.

Only as an example, during the implementation of IMAC processes, **some workflow patterns** emerged from user requirements:

- **The assign pattern**
- **The "take charge" pattern**
- **The warehouse transaction pattern.**

The assign pattern

A group supervisor can assign an activity to a specific member. The member name then appears in a column of the pending process list.

The "take in charge" pattern

When an activity is claimed by or assigned to someone, he must explicit signal to the rest of the group when he is effectively taking charge of the activity (a sort of read receipt from email). His name and the fact that he is now working in it appears in columns of the pending process list.

The warehouse transaction pattern

Every physical CI, which enters or exits the warehouse, must be processed by a piece of flow made by a couple of activities. One activity is in charge of the operation group, getting or handing over the CIs, the other involves the warehouse creek.

The two activities should be executed in strict sequence (possibly on the same PC), like a transaction, to guarantee the correct checking of the CIs both by the technical group and the warehouse creek. These transactions should eliminate (or limit) the sad cases, where the CIs, declared as given back by the technical team, are different from those effectively received by the warehouse and viceversa.

18 Supplier management

Contract management

The contract can be structured in many ways, but it is useful to define **a way to link the pricing to the expected results**, where a result could be a concept or an item typical of the platform (e.g. a process). **The item** corresponds to the final artifacts to be transitioned. Consequently the work(package) is structured on well defined packages and artifacts to be delivered and transitioned.

Every final artifact, to be transitioned, can be further classified in different levels of complexity, evaluated on objective criteria defined with the supplier, to guarantee an adequate effort compensation.

Every combination of item/complexity defines a global amount (which can be measured in man-days, man-months, or directly as a money amount).

As already stated, platforms drives. The artifacts should trace down to the minimal working item in the platform (report, process, a web page, ...)

In **case of mixed approach**, with part of the activities insourced and part outsourced on the same artifact, the effort, to produce every artifact/result, can be also **divided in well defined phases**. Every phase has specific entrance and exit criteria, consume and produce intermediate artifacts (e.g. analysis, prototype development, final development, test, transition, production).

Every item phase can be defined as a percentage of the item price. The sum of the percentages on all the work phases is 100%, that is the full amount. This way, it is possible to insource a phase and outsource another, defining reciprocal expectation between the provider and the supplier (e.g. analysis insourced, the rest of the phases outsourced).

This approach is completely different from a time and material. It is also different from "the Big One Project Objective", which would be very difficult for us to manage and control.

The **approach is similar to a menu pricing**: many targets, every one separately managed, controlled, evaluated, priced and paid when ready.

So you have better control on your achievement and your spending.

In this way the overall risks on the project, regarding both supplier management and results, are fragmented and reduced to the small dimension of a single item.

At the end, **you will spend the same amount of the bigger project approach**, but the effort and the expenses are simpler to justify internally and externally, and **a big project is partitioned in smaller, independent, singularly controlled batches**.

Adoption of remote collaboration tool and bug tracking

In the operations of the project, to reduce costs, the supplier worked in its location, quite far from ours.

Even if the work is structured and the analysis is completely in sourced, there is the need of frequent information exchanges and direct support.

To limit the inconvenience of distance and different locations, many **distance collaboration tools** have been adopted:

Conference call by Phone or Skype: to have "face to face" explanation sessions, from the supplier to the author of the analysis, from the internal team to supplier team, to define details through the live prototypes.

Team Viewer: this remote control tool was useful in situation where the supplier should give support on rare occasions of problems on the platform they could not reproduce in their systems; the adoption of Team Viewer means that a member of the internal team shares the desktop of his Pc (used as a bridge, due to internal security policy constraints), to let the outsourced team member get to the internal server.

Bug tracking tool in cloud (or as a service): to truck bugs, implementation requirements change and other implementation activities, a bug track platform as a service has been adopted; in this way the access is granted to every member of the dispersed team from every location.

Email or cloud storage: to exchange (small) files, the main tool is the email; a sort of "human interaction protocol" and naming convection on files has been defined, to track versions and to prevent problems like "which is the newest version of this file"; in case of very large file a cloud storage is used.

Processes and best practices: to proceed smooth, a set of practices (e.g. Style Rules) is implemented and constantly followed by all the members of the team, under the supervision of the project manager. As these pertain to the quality of the product, a specific paragraph is dedicated to them.

19 Project Templates

Templates, why

Starting the project, soon the need to guarantee tracing, uniformity and completeness of analysis arises.

One useful way to enforce this needs have been the adoption of document templates, in particular for processes.

In this chapter there are the most important we use.

Process Analysis Document Template

Process Name

Description

What the process do in synthesis

Involved Organizational Units

The list of OUs or a reference to the diagram

Process Activation Trigger

The event that triggers the start of the process

Process Ending Criteria

The normal conditions by which the process ends and the objectives of the process.

Data managed by the process

Information name (Field name)	Information description and further specifications

Additional specifications on data, e.g.:

- Functional dependencies among different data
- Set of values of a field and their meaning
- State machines

...

Documents managed by the Process

The process could produce a set of documents or form in pdf/paper format (E.g. a consignation form to be signed by the user).
For every document the following must defined

1. Name and type of document
2. Information to be put necessarily on the document
3. Information to retrieve "on the field"
4. A template of the document

In every process activity which manages this document (produce or acquire), the reference to the document must be added in the "Action on data/artifact" section of the activity.

Notification Template (via email)

The template of email with all the relevant information needed to complete the template in a final email mast be added in this paragraph and referenced in the "notification" section.

Policy to manage the risk of process interference on processes of the same type

Every process must be analyzed also in the presence of many instance of its type.

The problem is linked to the concurrent access of many processes to the same items, which is caused by a group of processes of the same type running concurrently.
The problem is both a transactional problem in DBMS terms and a practical problem in operational terms, e.g. in case of a massive assignment of Pcs. In this last case, as every Pc assignment is a process, from a DBMS point of view, when a Pc is linked to a process, it should be impossible to select it in another similar process.
From an operational point of view, what can happen in massive assignments is that the link between a Pc and a process could be purely casual. The final delivery of the set of Pcs will establish which Pc is assigned to which user. The process should be designed to manage also this particular situation that normally arises in day-to-day operations.

Activities

Every process activity is described by the following card, with many sections to fill by the analyst.

Where applicable, the various convections must be followed: naming, attribute ordering, ...

The process activity card is tailored on CMDBuild workflow feature.

Name	The short name of the activity, which is its Id
Description	Detailed description of the activity
Roles	The involved OUs; if there are many, the activity can be performed by more than one. Normally is one and is referenced the process diagram.
Conditions to carry on following activities	If there are many output transitions from an activity, every transition is conditioned and the couple (conditions with the target activity) must be listed.
Actions on Process Data/artifact/linked CI	Elencare un sottoinsieme delle informazioni di cui sopra, specificando se l'utente deve solo vederle, modificarle ho ha obbligo di modificarle Includere anche eventuali attività accessorie, come consultazione di dettagli di qualche informazione Specificare inoltre se deve essere prodotto e/o allegato un documento o mandata email
Automatic Updates and procedures	Specificare tutte le operazioni automatiche sui dati fatte dal processo
Applicative status of the process	Eventuale modifica dello stato applicativo del processo
Notification (email)	Nel caso ci fossero, va specificato a chi e quando (cioè quali condizioni, ovvero interattiva), il template della notifica va posto in apposita sottosezione del paragrafo documenti

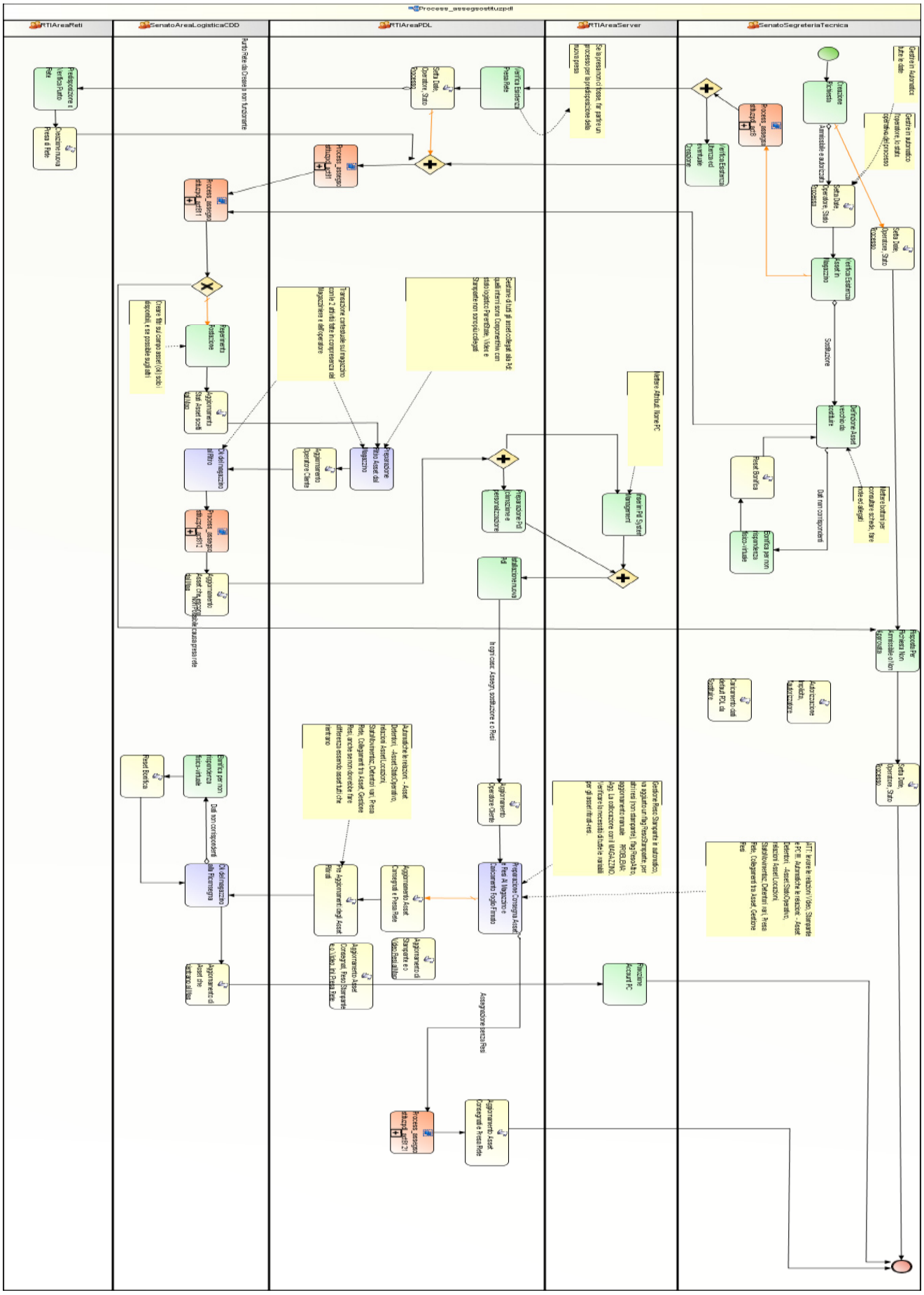
Exception management/Remediation of the Process

Here the list of possible exception situations is listed with the way they are managed in the process, both from an applicative point of view and an operational point of view.

Recovery or remediation mechanisms are also listed here.

It is of particular importance what to do in case the process is abruptly aborted to maintain the coherence of the data.

Process Flow Diagram



Test Plan Structure

A test plan document accompany every artifact, processes in particular, to drive tests and be sure to verify every relevant aspect.

For process, aspects are correct flows executions and correct data updates.

For processes, the Test Plan is composed of a set of test paths, drawn on the process diagram, with the expected results.

Test Result Template

Process Name:	Es. Assegnazione PDL
Processo Version	Es. V7
Test Date	

<Fill a card for every test scenario>

Test Scenario	**SCENARIO 1** Richiesta CPU, Video e Stampante, NESSUN INTOPPO

Note: put in **bold underlined style**, negative or doubtful results; in normal style, those ok.

Very negative, significant results and bugs can be put in dark red

ATTIVITA'	*RISULTATI*	*COMMENTI*
A<x> <Nome Attività>	<Se tutto ok mettere solo ok, solo per gli asset ad aggiornamento automatico esplicitare la verifica, specificando se prima o dopo il "Continua">	<Mettere qualunque osservazione/suggerimento/Commento sull'esecuzione del test> Es. Lentezza in fase di transizione
A2 Creazione Richiesta	In lavorazione **Email non inviata**	