

# Enterprise Architecture & Agility

White Paper

October 2008

The CEISAR produces White Papers on Enterprise Architecture every 6 months.

The themes are chosen by the Sponsors of the CEISAR: Air France, Axa, BNP Paribas, Michelin and Total. For the period April-October 2008 the themes dealt with were **Transformation Agility within the Enterprise** and the **Training of Business Analysts**.

This document summarizes the key recommendations defined by the CEISAR workgroup in which Sponsor representatives took part. For their contributions, we'd especially like to thank:

- Christophe Binot (Total)
- Marc Blangy (Axa France)
- Marc Blottière (Groupe Axa)
- Eric Boulay (Arismore)
- Jean-Baptiste Ceccaldi (Vistali)
- Raphaël Derbier (Vistali)
- Jean-Marc Guiol (Groupe Total)
- Jean Christophe Lalanne (Air France KLM)
- Pascal Léautier (Air France KLM)
- Guillaume Lemele (Axa France)
- Christophe Longepé (BNP Paribas)
- Laurent Mondemé (Air France-KLM)
- Lionel Pequignot (Total)
- Pierre Pezziardi (Octo Technology)
- Jacques Printz (Arts et Métiers)
- Danielle Rattier (BNP Paribas)
- Vincent Schattner (BNP Paribas)
- Philippe Tassin
- Bertrand Thyrion (Michelin)
- Michel Volle
- Denis Zandvliet (Value 360)

Responsibility for these recommendations resides with the CEISAR alone.

The first part of the document summarizes the recommendations for concrete actions to increase Agility, classed according to level of responsibility: General Management, Business Lines, Foundations Team, IT Department, HR Department, Purchasing Department.

The sections which follow examine these recommendations in greater depth.

## Contents:

1	What concrete actions to improve Agility .....	4
1.1	What General Management can do .....	4
1.1.1	Request a Transformation appraisal.....	4
1.1.2	Set up a "Foundations" Unit which works for the "Common Good" .....	4
1.1.3	Set two goals for the Foundations Unit: Agility and Synergy .....	4
1.1.4	Define the functioning of the "Foundations" Unit.....	5
1.2	What the Business Lines can do .....	5
1.2.1	Separate Operations and Transformation.....	5
1.2.2	More in-depth choice of Software Packages.....	5
1.2.3	Solution-Project Team cooperation with the Foundations Team .....	6
1.2.4	Appoint Project Leaders to both Manage and Build .....	6
1.3	What the IT Department can do.....	6
1.4	What the Foundations Team can do.....	6
1.5	What the HR Department can do.....	7
1.6	What the Purchasing Department can do .....	8
2	The CEISAR Model and Transformation vocabulary .....	9
3	Agility: a key competitive edge .....	11
4	Processes and Functions .....	14
4.1	Progress in Project Management .....	15
4.2	What Functions for Project Engineering? .....	15
4.3	Engineering improvements to enhance Agility .....	18
5	A single model shared by Business and IT .....	20
6	Defining and sharing Business language.....	22
7	Reuse of components.....	23
8	Business-Process and Organized-Process.....	27
9	Contractual Approach or Cooperative Approach.....	28
10	The Business Roles of Transformation .....	31
11	Organization .....	34
12	Training of Business Transformers .....	36

---

# 1 What concrete actions to improve Agility

Changing Operational-Processes via a Transformation-Process is a difficult task.

Changing the Transformation-Processes themselves in the aim of making them more efficient is even more difficult because they are **more complex**.

It is not possible, in large organizations, to rapidly apply the recommendations we will here describe. It requires a **progressive** approach. The series of recommendations we will make here deeply modify both Organizations and Methods, in particular:

- by introducing a **Foundations Team** whose mission is to increase Agility and Synergy
- by moving from a Contractual Approach to a **Cooperative Approach** which brings together Business and IT in the same project team
- by encouraging Project Leaders to position themselves not simply as **Managers** of the project but also as **Builders** of Enterprise Solutions

We also recommend that each Enterprise tests recommendation viability by way of a **Pilot Project** prior to widespread deployment.

## 1.1 What General Management can do

### 1.1.1 Request a Transformation appraisal

General Management usually identifies scope for progress in Operations such as optimization of the Supply Chain, or the Sales Processes, or Back Office Processes. Means are then budgeted to improve these Operational-Processes. However, General Management tends to allocate little resources to the improvement of Transformation-Processes. Firstly, because it is rarely aware of the overall cost of Transformation and secondly because it does not believe that it is possible to strongly progress in this domain: such projects are risk-laden, there's not much we can do.

It is thus necessary to carry out:

- An appraisal of what Transformation **costs** to the Enterprise: not just the cost of IT Development, but also what it entails in terms of the Business Actors, the Transformation tools, the management and associated governance costs, the training for Operational Actors, the doubling up of certain tasks during initial Solution deployment, the data migration...
- An appraisal of the degree of satisfaction with regard to the **deadlines** for the creation or modification of Solutions and their **quality**.

Such an appraisal will inevitably show that both the overall cost of Transformation and the frustration of the Business Lines with regard to Solutions Agility require that the problem be seriously addressed.

### 1.1.2 Set up a "Foundations" Unit which works for the "Common Good"

Each Business Line is judged on its efficiency. It is thus illusory to ask it to work from the "common good" of the Enterprise, its behavior is inherently selfish. This common good is the responsibility of ...the General Management. To meet this responsibility it must set up a **"Foundations" Structure** which works on the common behalf of the Business Lines and which regroups both Business and IT skill excellence.

### 1.1.3 Set two goals for the Foundations Unit: Agility and Synergy

The role of the Foundations Unit is to enhance **Agility** in both of its aspects: "fast" and "well", and to identify the right level of **Synergy** between Business Lines (or Subsidiaries).

The "Methodology", "Enterprise Architecture", "Enterprise Glossary", "Security", "Development Tools", "Technical Infrastructure", "Interfaces" or "Components" teams ought not be dispersed... There exists but a **single**. "Foundations" Unit responsible for all that is mutualizable between Business Lines so that the Solution Project Manager only has to deal with a **single** internal supplier whose role is:

- To build and support a unique range of **Processes** and **Transformation Tools** reusable by all the Business Lines Transformation teams.

- To build and support **Models** that are **Reusable** by the Business Lines: definition of a common Business vocabulary to facilitate Business/IT dialogue, organization principles, security norms, user interface standardization, Reusable Components, and IT Infrastructure...
- To establish measuring instruments to monitor the evolution of **Agility indicators** (rapidity of change and quality of Solutions) and of **Synergy Indicators** (sharing of systems of reference, reuse of Solutions or Components).

#### 1.1.4 Define the functioning of the "Foundations" Unit

- Appoint a skilled **Foundations manager** recognized by the Business Lines: his or her role is an extremely difficult one.
- Ask him or her to define an **action plan** that is **adapted** to the Enterprise: Synergy needs are different in Industry and Services. Deduce the appropriate means to be allocated to this structure.
- Establish **re-invoicing** rules which do not dissuade the Solution teams from reusing the Foundations.
- Define rules of **Governance** which lead the Business Lines to reuse what is offered by the Foundations team: Checking of conformity of Solution Projects vis-à-vis Foundations must take place **before** validation of the project and the budget (see white paper on Governance); if not, the Solution Project Manager will explain that the reuse of Foundations is incompatible with the planning and budgets already decided upon.
- Define rules which lead the Foundations Team to behave as a **Supplier** to the Business Lines and not as a hierarchically superior structure.
- Encourage the Foundations Unit to **recuperate** and package the Components deriving from the Business Lines projects.
- Monitor the **progression** of the agility indicators.

## 1.2 What the Business Lines can do

### 1.2.1 Separate Operations and Transformation

Short-term concerns always win out over long-term issues: a functioning hitch in Operations requires a **priority** reaction with regard to incidents which may affect the Transformation Projects under way. If a Business Line wants to enhance Agility it must therefore imperatively **separate** Transformation responsibilities from Operational responsibilities.

### 1.2.2 More in-depth choice of Software Packages

A readily available Solution is always more attractive than a Solution that has to be built: Hence the success of software packages.

- Business managers, when choosing, can envisage an already available **concrete** Solution that offers a large part of the desired Processes. If the Business Lines have suffered setbacks in the past due to poorly handled projects, they can feel **reassured** to be able to rely on a Solution that is already successfully running in other Enterprises.
- Business managers perceive their **task of defining new Processes** as **facilitated**.
- **Costs** and **deadlines** should be reduced compared to an independent Solution because the investment and evolution costs of the Software Package are mutualized between several Enterprises.

What we call a **Commodity Solution** is a Solution whose needs are **foreseeable** because they have to do with Production, Back-Office, or regulation... A "Commodity Solution" will of course evolve in keeping with changes in organization and in regulations, but it will evolve within a known perimeter.

What we call a **Competitive Solution** is a Solution whose needs are **not foreseeable**, such as: CRM, Business Intelligence, partner relationship management, Product Factory... Through Solution use, Operational Actors progress their Solution in directions they cannot predict a priori.

For each Solution, objectively weigh up 2 scenarios, **Software Package Solution** or **Specific Solution**: one must take into account not just the available Functionalities but also the costs and overall deadlines

(in particular, for future versions), differentiation possibilities, speed of evolution, insertion of Software Package Architecture into the Enterprise Architecture (duplication of data, specific user interface, specialization of Transformers...).

**The Software Package Solution can only constitute a response to Competitive Solution needs if it is built from reusable Components.**

### 1.2.3 Solution-Project Team cooperation with the Foundations Team

- Work in a spirit of **cooperation** with the Foundations Team: it must not be considered as a General Management watchdog but as a contributor to be listened to and solicited. **Use positively** what the Foundations Team delivers: before going specific, begin by **reusing** existing Components, even if it involves asking for improvements or complementary Components.
- Act as a **proposition** maker vis-à-vis the Foundations Team. Know how to "lose time" at the end of each Solution-Project to **recuperate** what can become reusable Components for other Projects. In general, Foundations Team intervention is necessary to package the Component prior to roll-out to other Business Lines.

### 1.2.4 Appoint Project Leaders to both Manage and Build

Choose and Train Project leaders who are not simply **Managers** but who are also **Builders**. Protect Project Leaders from an excess of administrative tasks.

## 1.3 What the IT Department can do

The IT Department must back its Foundations Team: see the following paragraph.

Furthermore, the Group IT Department, by virtue of its transversal role, must support General Management in its quest for the Common Good: it has a key role in defining the right level of Synergy between Business Lines to promote economies of scale, to communicate and to train.

It must also clarify **outsourcing** choices for IT development: these are based on lower cost of manpower, but they can hinder the Cooperative Approach and often lead to a loss of knowledge of the Enterprise Model.

It must also fight against excessive "proceduralization" which leads to believe that good Project Management guarantees good Solution Building.

Just because "quality", "security", "ergonomics", "organization", "method", "integration" "performance" and "tests" are important, doesn't mean we need to define one Role per topic. **Multiplication of Roles** increases problems of follow-up, coordination and integration, and it **de-responsibilizes** the teams. Instead seek out quality players **capable of taking on several roles**.

## 1.4 What the Foundations Team can do

- Behave with regard to the Business Lines as a **supplier** vis-à-vis its clients and not as a higher authority.
- Define the "Solution Building" **approach** which recommends the following points:
  - **Group** Business and IT Builders together into a **single Project-Solution Team**
  - Define a **single** Approach shared by Business and IT, something which is not yet the case in all Enterprises.
  - Favor the **Cooperative Approach** whenever possible: rather than an **exhaustive** inventorying of needs, favor the Building of a Solution Architecture able to easily accept successive additions according to the gradual maturing of needs.
  - Check that the **Problem** is well defined: so that all Project Actors take the right daily decisions, they must systematically refer back to the Problem to be solved, which presumes that everyone has grasped and integrated the Problem so as to properly **align** the Solution to it. Thus it is important to provide a **simple explanation (1 page)** of the Problem in hand and to communicate it to all, for instance by posting it up in the Transformation team offices.
  - Draw up a **rigorous glossary of Business Entities** reusable from one project to another. Ask all those involved to use and gradually fine tune it. This Glossary is **extended by the Business Entities Model** (relations, inheritance, identifiers, Entity life cycle) which represents the base Solutions Architecture.

- **Dissociate** Core Business and **Organization** in the analysis of Processes so as to build long-term Solutions which support successive Organizations.
- Look for "Round-Trip" Modeling **tools common** to both Business and IT (for Data Models, Process Models, Rule Engines, User Interface): we Model the Business and it automatically generates part of the Software; we Modify the Software and it automatically updates the Business Model.
- Manage **reusable Software Package Solutions** between the different Companies of a Group: choice, follow-up of Versions, tool building for Interfaces and Migration.
- For **Specific Solutions**, progressively build **Reusable Components**
  - Set oneself a **sizeable** goal: even if one starts small, one can achieve reuse rates of up to 70% for Specific Solutions
  - Take into account **all** types of Components: access to Data, flow between Solutions, SOA Components, patterns, use of Rule Engines, workflow engines...
  - Acquire **know-how** in Component building
  - Make Components easily **accessible and understandable**
- **Support** Reusable Components with regard to Solutions Builders in the spirit of a software editor and not a "guardian of the temple"
  - **Train and Help** (coaching, assistance, reviews) Solutions Builders: Business and IT
  - Host a **hotline**
  - Gather **requests** for correction and evolution
  - **Observe** the use of Components to concretely improve their practical use
- Define the **road-map to simplify the overall Enterprise Architecture**. A project will be simpler if it is part of a well structured Enterprise Architecture. The clarity of the perimeter, the precision of the interfaces with other Solutions, the reuse of Data access Functions are key advantages to helping Project Managers to focus their energies on the Solution Model and not on its environment. The difficulty resides in establishing a progressive simplification strategy so that each Project contributes to this overall simplification. This approach has already been defined in a CEISAR white paper called "**Simplify Legacy Systems**": we invite readers to download it at [www.ceisar.org](http://www.ceisar.org)

## 1.5 What the HR Department can do

- If we want to assign talented people to **Business Transformation**, we must **valorize** (salary, training, recognition) **the Transformation function** and not define the size of the (Operational) teams managed as a key criterion of recognition.
- **For** "Business Solution Building" **select** Actors of high quality, capable of carrying out a maximum of Transformation tasks, to avoid multiplication of Roles (see above).
- Offer adapted **Training** to these Business Builders to enable them to rapidly mature:
  - To understand what **Enterprise Architecture** is: alignment with Strategy, break down into Operations and Transformations, break down into real World and Model, the quest for Synergy via the Reuse of Models and the Sharing of Resources
  - To understand what an **Enterprise Model** is: Data Model, Action Model (Process and Functions) and Actor Model (human Actors and computers)
  - To know how to execute a project: **break down into 8 Engineering Functions** (Understanding the Context, Defining the Problem, Building the Solution, Checking the Solution, Adapting the Solution, Configuring the Solution, Deploying the Solution, Operating the Solution).
  - Understand **Cooperative Approach** and **Contractual Approach**
  - **Personal skills**: prepare the Business Builders to separate out the essential, structure from detail, and not count on an "unquestioning" respect for Transformation Process to make their project succeed.
  - **Knowledge of Foundations**: one must first explain that it is possible to Build Solutions for highly **specific** needs with the help of **common** Foundations. Once this principle has been acquired, the variety of Components that one can reuse ought to be demonstrated.

- Detail each **Engineering Function** in terms of its Business component: define the Problem, build Business Entities, Functions, Processes, check, configure, train...
- A good Project manager ought to have qualities and skills as both a Manager and a Builder. Management skills have been valorized more because they sufficed for a Contractual Approach adapted to **Commodity Solutions**. Yet today they are insufficient for the application of a Cooperative Approach adapted to **Competitive Solutions**: the quality of the Solution Architecture is key. As good Builders are harder to find than good Managers, we recommend that for Competitive Solution "Project leaders" it is preferable to choose the **best Builders** rather than the best managers per se, even if it means supporting the Builders with people in charge of Project administration. This represents a key cultural change in Organizations which have tended to highlight Management Roles.
- Maintain a core of top flight **IT skills**, which will enable the company to conserve the knowledge of its Model. Increasing outsourcing of IT Modeling will hinder the implementation of a Cooperative Approach in Enterprises. It is recommended that all Enterprises conserve a **core of proven IT skills**. If not, the Enterprise will one day find itself dependent on suppliers who have evolved their IT Solutions while the Enterprise has no longer conserved knowledge of its Model: the original Business Model has not been updated and has lost its value, while the **Software** has been successively updated by the Supplier who alone understands it.
- Favor **bridges** between the Foundations Team and the Business Lines.

## 1.6 What the Purchasing Department can do

- Underline **Quality** and not just price in the selection procedures for suppliers who contribute to Transformation: as opposed to Operations, productivity disparities in Transformation can range from 1 to 10.
- Favor partnerships with small **innovative** structures. The larger suppliers often have Approaches which they have tailored to **comply** with what big companies "want". Certain small atypical suppliers can breathe new life into things.
- Find new forms of partnership:
  - **contracting** based on time and money is not very satisfactory: one buys human resources at the lowest cost, independently of the result, which has led certain sponsors to proscribe this kind of partnership
  - the **fixed price project** is no longer satisfactory: it requires the modeling of **all** business demands prior to contracting. This is increasingly difficult when moving from Commodity Solutions to Competitive Solutions, whose perimeter is variable and which will require a Version by Version approach.
  - One must envisage partnerships based on the following principles:
    - the Solution must be **well built** so as to accept subsequent add-ons
    - cost and deadline are controlled for each successive **Version**: this could take the form of a succession of small fixed price projects
    - the functional perimeter of each Version is broken down into a compulsory part and a **variable** part gradually adjustable according to project advancement, as long as costs and deadlines are well respected.



---

## 2 The CEISAR Model and Transformation vocabulary

The CEISAR has defined an Enterprise Model (see [www.ceisar.org](http://www.ceisar.org)) which can be outlined as follows: **Enterprise Architecture** defines how an Enterprise **Operates** (the "present") and **Transforms** itself (the "future"): how **Actors** (Humans or Computers) carry out **Actions** (Processes, Functions) with the help of **Information**.

The Architecture does not define the **strategy** of the Enterprise (what products, for what markets, with which partners, on which territories), but this strategy is an "input" for the Enterprise Architecture.

When the functioning of the Enterprise becomes too complex its **Operations** have to be **Modeled** so that they can be carried out efficiently: the Model enables us to understand, to memorize, to communicate, to train, to guide Actors.

Modeling can be **global**, via Maps (Business Entity maps, process maps, Function maps, Block maps, Services maps...) or **detailed**.

The detailed model comprises 3 parts:

- The **Actor Model**: the Human-Actor Model is called "**Roles**" (Vendor, Producer, Administrative), the Computer-Actor Model is called "**IT-Configurations**" (Hardware, Software, network).
- The **Action Model** describes the list of instructions to correctly execute an Action: it is a **User Guide** for Human-Actors and a **Software** for Computer-Actors. We distinguish:
  - the **Process Model** ("Sell", "Produce", "Manage")
  - the **Function Model** made up of the Processes ("Pricing", "Print")
- And the **Data Models** for Customer, Product, Contract, Account...

### **Business = Core Business + Organization.**

We call "**Core Business**" all that defines the Business independently of the Organization chosen by the Enterprise: definition of Products (Goods or Services), Customer Data, choice of Partners, pricing rules. We call "**Organization**" the organization chart, the Role of Actors, authorizations, duties, the distribution of Activities.

This fundamental distinction ought to help to build a Solution which is based on the Core Business and supports the different Organizations (successive or parallel).

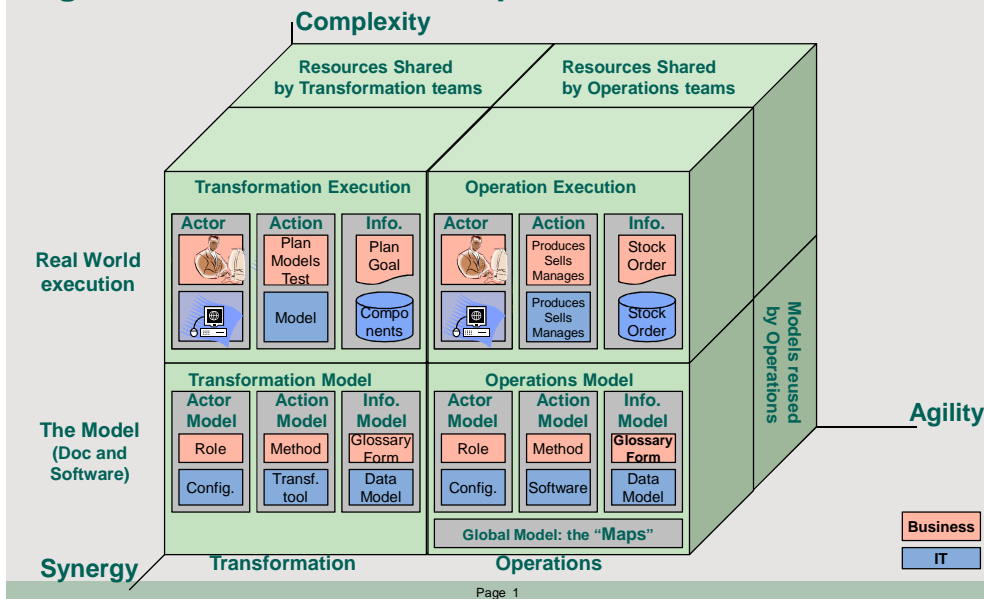
The Enterprise must evolve its Operations Model to contend with ongoing change.

The Building (creation, acquisition or modification of an Operations Model) and its Deployment are called **Transformation**. It is no longer a matter of selling or producing, but of managing Projects, of defining a "road-map", of changing a price, of modifying a Process, of opening an agency...

Just as Operations are executed thanks to an Operations Model, Transformation is executed thanks to its **Transformation Model** which describes:

- The **Actor Model**: Roles of "Project Leader", "Architect", "Project Manager" for Human Actors
- The **Action Model**: **Process** model (i.e.: "Build a Solution") or **Functions** Model which make up the Processes ("describe a Problem", "Analyze a Process", "Test a Function")
- And the **Information Models** (Project, Test, Planning)

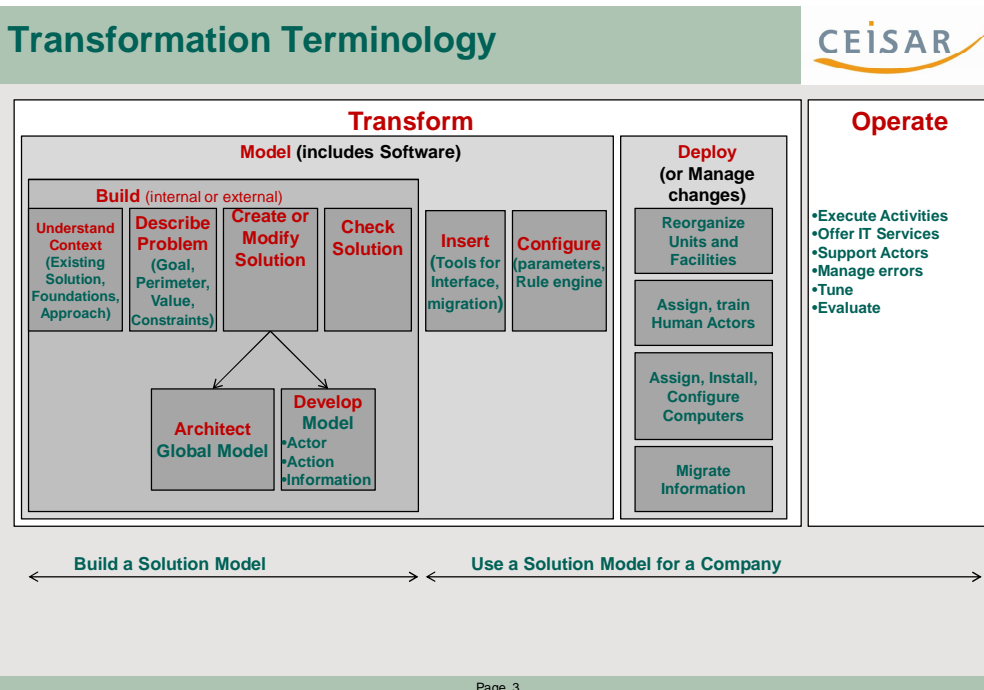
# A Single Enterprise Model for Business, Organization and IT with adapted views



The CEISAR Cube is a framework whose 3 dimensions correspond to the 3 major challenges that Enterprises face today: **Complexity**, **Agility** and **Synergy**.

- managing growing Complexity requires separating the real world from **Modeling**
- managing Agility requires separating Operations and **Transformation**
- managing Synergy requires **Sharing** Resources (human resources, IT resources or data repositories) and **Reusing** Models (Solution Models or Components or Approaches)

In this White Paper, we are focusing on the left-hand part of the Cube: Transformation.



Transformation includes **Modeling** and **Deployment** of the new Model in Operations  
 For example, opening a new branch with the help of an already existing Model is part of Transformation because it is a matter of Deployment of a Model.

Modeling includes not just the **Building** of the Model, but also the **Insertion** of the Model into the Enterprise Architecture (Interfaces and migration tools) and the **Configuration** of the Solution with the help of parameters and rule engines.

---

## 3 Agility: a key competitive edge

**Reactivity** = Operate fast and well - and **Agility** = Transform fast and well.

Rapidity in serving one's customers, rapidity in turning over one's stock, rapidity in paying one's suppliers... this is what we call "Operations Reactivity": a company is reactive if its Operation Model enables it to react fast and well to the demands of its Customers, Partners and Suppliers...

Yet the Model itself has to be transformed fast and well, which is no longer a matter of Operations but of **Transformation: Reactivity is the art of Operating fast and well** while **Agility is the art of Transforming fast and well**.

### Fast and well

To respond to the **Problems** of the Enterprise, we create or modify **Solutions**: how Human **Actors** or Computers must **Act** with the help of **Data**.

The creation of Version 1 of the Solution is more costly than each of the successive Versions because it includes the Building of the Solution **Architecture** which will be conserved for the following Versions. Yet given the life-span of Solutions, the sum of the cost of Versions 2 to "n" represents 2 to 3 times the original cost of Version 1. Thus, Agility is not simply the art of **rapidly** building the first version of the Solution, but also the art of building in, as of this first Version, an **efficient** Solution Architecture which subsequent versions can avail of. Agility at instant T ought not to compromise agility at instant T+1: this constitutes the definition of sustainable development as applied to IT Systems.

### Transformation is more complex than Operations

We know how to Model the "Order" Operational-Process, but we don't really know how to Model the Transformation-Process "Build a Solution".

The uncertainties are many:

- Uncertainties concerning the deliverable
- Uncertainties concerning the Architecture of the Solution
- Uncertainties concerning the development of the Project
- Uncertainties concerning the level of acceptance by Operational Actors.

These require that the Project manager have the necessary talents to take decisions in an uncertain environment.

We have gradually succeeded in Modeling the **Management** Functions: Governance, planning, budget, resources, communication...

We have not yet really succeeded in Modeling the **Engineering** Functions: how to define the Problem, how to Architecture the Solution, how to reuse Components, how to Build a Solution that supports different Organizations.

### Agility = a key quality of an Enterprise

Agility is sought after by all Enterprises because they have to face an increasingly shifting environment which obliges them to constantly change.

As it is impossible to guarantee that the innovations initiated by an Enterprise will be better than those of its competitors, it would seem that the surest strategy to grow efficiency is not just to innovate in-house, but above all to develop Transformation Processes which are more **agile** than those of one's competitors, so as to rapidly implement innovations from elsewhere and to rectify weaknesses faster than one's competitors. In this respect Agility has become **the key quality of an Enterprise**: If productivity, quality or reliability is poor, if products are too classic or too expensive... Agility allows rapid improvement of such weaknesses. They only have to be identified and Agility enables fast reaction.

If a Large Enterprise is agile, it can acquire start-ups, copy successful innovations by certain competitors and obtain a genuine competitive edge **not because is it "far-seeing" but because it is "fast-moving"**.

Strategy is no longer based on uncertain forecasts of the future, but on the grasping of opportunities which the Enterprise can rapidly take advantage of, thanks to Foundations which make it more agile than its competitors.

## To equip itself with Commodity Solutions, Enterprises have developed a Contractual Approach which has given precedence to security or reliability over Agility.

We began by rolling out "**Commodity Solutions**" which automated well defined Processes for Production, back office, and regulation, the key objectives being productivity, reliability and visibility. Enterprise Transformation Processes adapted to Commodity Solutions are not always sufficiently agile, often because they were defined at a time when pressure wasn't as great, when needs were more easily foreseeable, when emerging technologies required strict controls, or when the use of IT development tools required that all needs be defined prior to taking action.

This slowness of internal Transformation, which leads to an increase of associated costs, is characterized by the drawing up of a Contract which contains the Business Model to be translated into an IT Model, the separation of the Business and IT teams, and a multiplicity of Roles whose relations have to be managed.

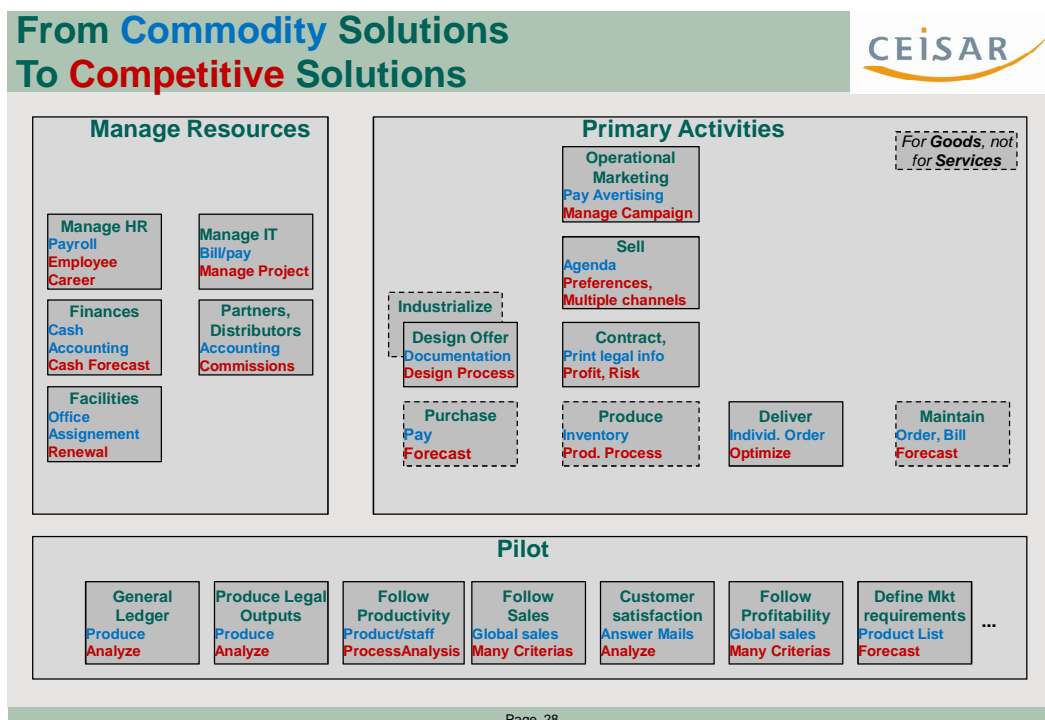
This cumbersomeness has enabled a **Software Package** industry to rapidly develop, by presenting itself as an alternative to Specific Solutions. Thus ERP has answered the needs of finance, HR, production management, back office productivity, reporting, and so on... Because the commodity needs were not very different from one company to another, it was thus possible to build a Solution common to several Enterprises.

## The Contractual Approach is no longer adapted to new Competitive Solutions

We are gradually switching to "**Competitive Solutions**" which can give an Enterprise an added competitive edge. Business Intelligence and decision aids, CRM and new front-office tools, product design and time-to-market, end-to-end process and integration of partners or customers ("The Extended Enterprise") multiplicity of distribution channels are just some of the domains in which Enterprises want greater agility.

They are characterized by the impossibility of defining all needs up front. This requires:

- a **new approach**: do not oblige Business to specify all; proceed by successive versions; put the emphasis on the quality of the Solution Build so that it accepts later Versions.
- a **new form of Reuse**: no longer reusing software-package Solutions that we can adapt on the margins, but reuse of Components that we can assemble to build innovative, differentiating Solutions.



Yet surprisingly, the very same Enterprises who were able to substantially optimize their back office or production **Operation Processes**, cannot envisage making strong progress in agility and many do not envisage re-founding their **Transformation Processes** that they had such trouble to have respected.

The aim of this document is not so much to make an appraisal of current Transformation Processes - on which much has already been written - but to persuade that it is possible to make progress, either by gradual actions, or by a "leap forward" so as to be able to build Competitive Solutions more rapidly.

## 4 Processes and Functions

### Different Transformation Processes

In parallel to Foundations Projects and Solution Projects, one of our Sponsors has created a category called "Innovating Project". These Projects comprise a greater share of **risk** and cannot succeed if they have to strictly apply classic Transformation Processes. What's more, these are Projects which are generally sidelined by classic Governance Processes which do not favor risk taking (see White Paper on Governance).

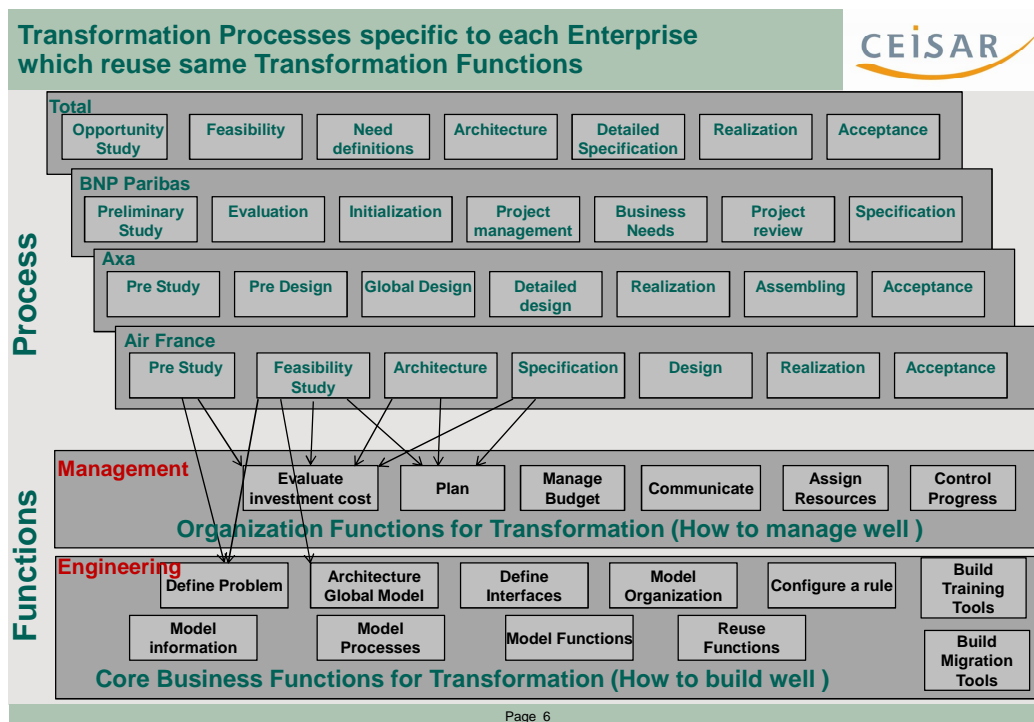
Today, 10% of their Projects are innovating Projects. They proceed via a different type of governance and apply more flexible Transformation Processes

Thus, there exist **different Transformation Processes**: Build a Solution, Build an innovating Solution, but also Modify a Solution (evolution or "debugging"), Build Foundations, Define a Road-Map,... In the main, this document deals with the Process of Building or Modifying of Solutions.

### A Transformation Process is broken down into Phases

We know how to successfully formalize **Operational**-Processes like the order Process and little by little they have been automated; this is much more difficult for **Transformation** Processes such as the Solution Building Process, given the uncertainties inherent to all projects.

However, many advances have been made: each Enterprise has defined its own Transformation Process by breaking it down into progressive **Phases**.



### Each Phase calls upon Functions

Each Process Phase uses Transformation **Functions**. For example, Air France's "Pre Study" Phase uses the Functions: "Define the Problem" and "Evaluate the investment cost".

Each Function can be activated **several times** in the same Process: for example the "Evaluate the investment cost" Function is fine tuned during each Phase to give ever more precise results. Similarly, the "Model Processes" Function is used at several stages to gradually detail the Processes concerned.

### Engineering Functions and Management Functions

We distinguish two types of Functions

- **Engineering Functions** (such as "Model Processes") to correctly **Build** the Solution
- **Management Functions** (such as "Plan") to correctly **Manage** the Project.

The Engineering Functions are the Transformation **Core Business Functions**: they must be carried out whatever the Organization in place. The Management Functions are the Transformation **Organization Functions**: they closely depend on the Organization. They are thus **specific** to each Enterprise, whereas Engineering Functions are **universal**.

## 4.1 Progress in Project Management

Many projects suffer from poor project management: the Phases are not formalized, the deliverables are not supplied, the planning is incomplete, the allocation of Actors is not anticipated, reporting is forgotten, decisions are not formalized...

Methodology definition bodies such as CMMI or Open Group (Togaf) have developed Transformation Management Functions in considerable depth. Enterprises have made tangible progress by taking advantage of this trend and by increasingly better mastering Project Management. We have gone from a period of improvisation to a period of continuous improvement, as is well illustrated by the CMMI levels of maturity which define how an Enterprise progresses in its project management.

According to study results published by the Standish Group (<http://www.standishgroup.com>), the success rate for IT projects has been rising steadily over the past 10 years. Indeed, 35% of projects are qualified as successful: they are delivered on time, within budget and to original specifications. However, half of projects are qualified as partially successful: they are delivered and operational, but have fewer functions than planned and are over budget and deadline. Yet the good news is: the number of projects qualified as failures – that is, abandoned while under way or with delivered results that are never used – is falling: it is now less than 20%.

The statistics for the past 10 years are as follows:

1995: Success 16%; Partial Success 53%; Failure 31%  
2000: Success 28%; Partial Success 49%; Failure 23%  
2004: Success 29%; Partial Success 53%; Failure 18%  
2006: Success 35%; Partial Success 46%; Failure 19%

In the main, Progress has been made in **Management**. By way of example, the Standish Group recommendations are essentially good Management recommendations:

- Commitment of top management
- User involvement
- Project manager experience
- The formulation of business goals
- Scope limited to essential needs
- A normalized technological infrastructure
- Precise and stable specifications
- Proven formal methodologies
- Rigorous and reliable estimates
- Others: splitting of deliveries, competence of personnel, etc.

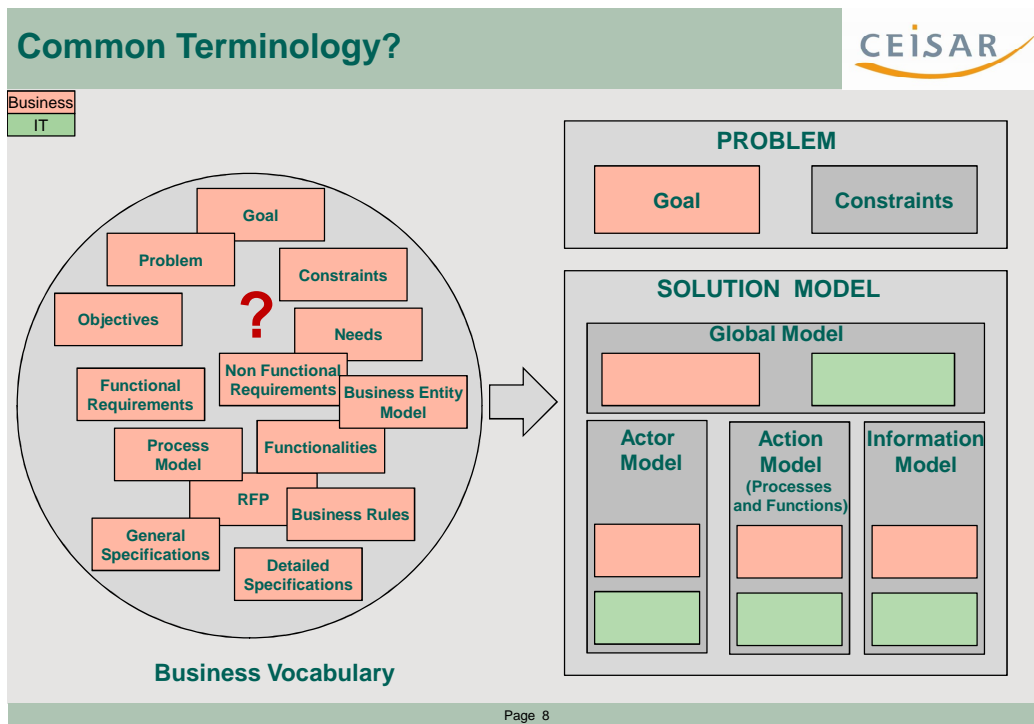
We must continue in this direction to improve Management, yet without overdoing it: too many Management tasks prevent the project manager from focusing on Engineering Functions. Don't forget, it is possible to **correctly Manage** the project of a **badly built Solution**.

## 4.2 What Functions for Project Engineering?

We will now offer a few ways in which Project **Engineering** may be improved, as today, it is a field that is less developed than that of **Management**.

Let us begin by defining the **Engineering Functions** classed from 1 to 7 (see further on): they describe the **Engineering Process** and not the **Organized Process**. For instance, one thing we do not see appear is the **Preliminary Phase** which triggers the Functions "Describe the Goal" and then "Architecture Global Model", nor the Feasibility **Phase** which triggers and develops the same Functions: so, Engineering Functions are represented only once, even though they are triggered several times.

The Transformation Process and the Transformation Engineering Functions are **common** to all Enterprises.



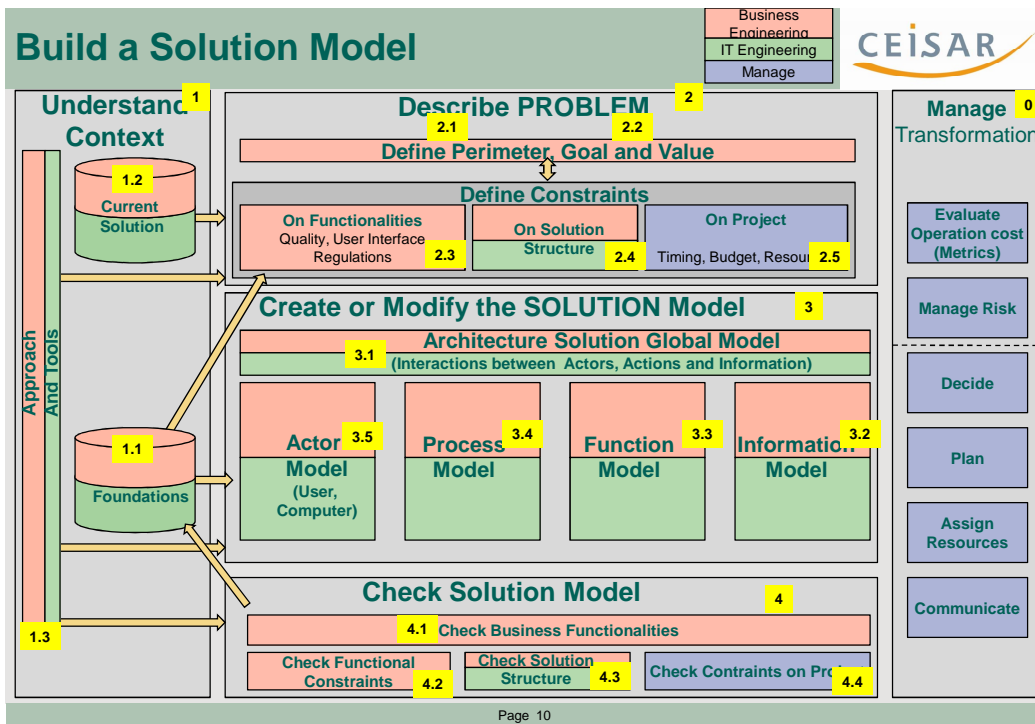
The terms "Objective", "Goal", "Need", "Request", "Functionalities", "Specification", "Design" are reclassified either into "Problem" or "Solution":

- **Problem:** here we find Goal, Objective, Constraints, defined by the Sponsor who decides and pays
- **Solution Model:** here we find what is Built in response to the Problem: that is, the Global Model, the Action Models (Process and Functions), the Information Model, the Organization Models, both for the **Business** part and the **IT** part.

Thus, the Engineering Process does not take sides in whether the Business and IT models ought to be separated or not: Contractual Approaches distinguish them, whereas Cooperative Approaches join them (see further on).

By way of example, the **definition of a new Process** is not part of the Problem, it is part of the Solution: in the diagram below, it is represented by the pink section of 3.3; the green section represents its IT translation.

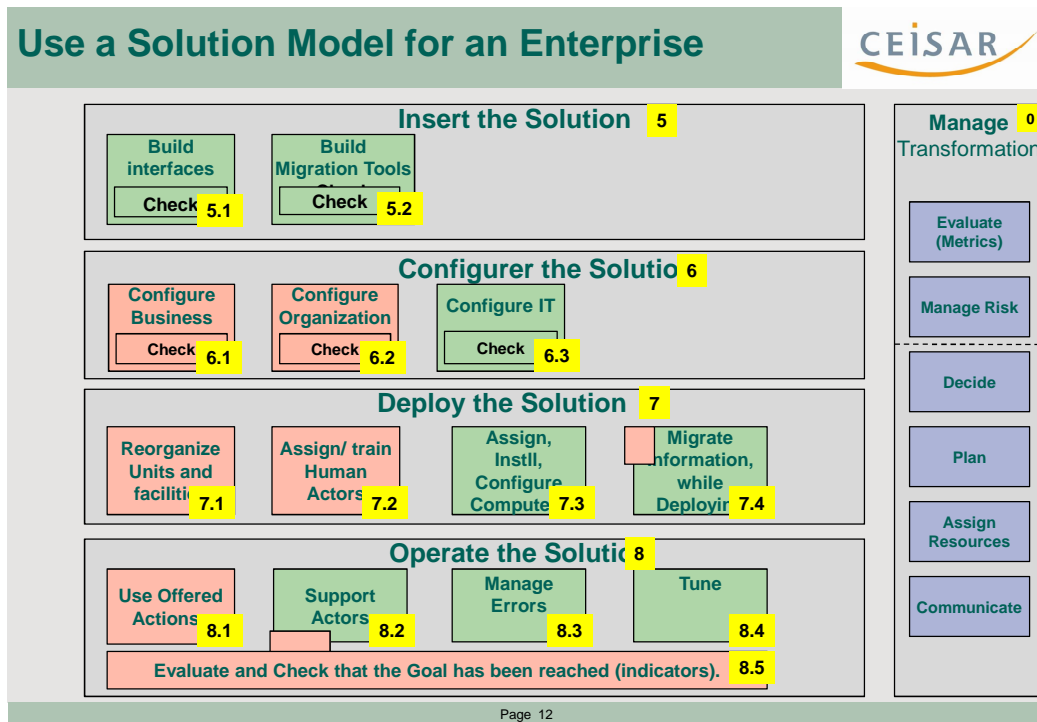




Page 10

1. Understanding the **Context**:
  - 1.1. Understanding the reusable **Foundations** to reduce Project workload and standardize the Solution usage. The Foundations include the Global Model (what is often called "Solutions Mapping") and the detailed Models (reusable Components, reused Information Models...).
  - 1.2. Understanding the **Solution to be replaced**: if it has to be replaced, there is no need to understand its internal Architecture, yet it is necessary to understand the Functionalities offered so that the new Solution re-includes them, and how it communicates with other Solutions
  - 1.3. Understanding the **Approach** and the **Transformation tools** recommended by the Enterprise
2. Describing the **Problem** to be solved:
  - 2.1. Defining the **perimeter** (geographic, Product, Process domain, customer segment...),
  - 2.2. Defining the **Goal** to be reached in business terms like "a gain of 10% in productivity", "launch a new range of Products", "reduce time-to-market", "organize a new partnership with a distributor", "launch a foreign subsidiary"
  - 2.3. Defining the Value **indicators** to check the result obtained at project-end
  - 2.4. Defining Functionality **Constraints**: regulatory framework, volumetry, performance, ease of use necessary to encourage utilization, etc.
  - 2.5. Defining Model Structure **Constraints** and its Interfaces with the other Solutions
  - 2.6. Defining the Project **Constraints**: deadline, maximum investment budget, resources policy
3. Create or Modify **Solution Model**: includes both **Business Model** and its translation into **IT Model**:
  - 3.1. Building the **Global Model**: define the interactions between Actor Models, Action Models and Data Models
  - 3.2. Building the **Actor Model**: Organization, Roles, allocation of Activities to Roles
  - 3.3. Building the **Process Model**: defining the Core Business Processes, breaking them down into Core Business Functions, then, defining the Organized Processes and the Activities
  - 3.4. Building the **Functions Model**:
  - 3.5. Building the **Data Model**: Business and IT definitions of Entities, Relations, Inheritance, Attributes, Types
4. **Checking** that the Solution Model solves the Problem (and that the Problem hasn't changed in the meantime):
  - 4.1. Checking the Functionalities offer (the "Business Action Model")
  - 4.2. Checking that the Functionality Constraints (2.4) have been properly respected
  - 4.3. Checking that the Model Structure Constraints (2.5) have been properly respected, and recuperate the Components that are reusable for other Models
  - 4.4. Checking that the Project Constraints (2.6) have been respected

These 4 Functions enable the Building of a Solution Model. This is the task that should be carried out for a Solution specific to an Enterprise, or by a Software Package manufacturer, or by a Group building a Solution destined for several subsidiaries.



This Solution Model now has to be **installed** in one or several Enterprises.

The following 4 Functions (Adapt the Solution, Configure the Solution, Deploy the Solution, Operate the Solution) can apply as many times as there are Enterprises which **reuse** the same Model.

5. **Insert** the Solution into the Enterprise Architecture: Build the **Interfaces** with the other Solutions and the **Migration** tools to import Information from the former Solution
6. **Configure** the Solution by way of parameters or a Rule engine so that it respects the characteristics of each Enterprise. Though it is a modification of the Model, it is perfectly localized and does not require complex integration Functions, or non regression tests ...Configuring a new Model ought to be simpler than Building it: indeed, ease of Parametering is a telling evaluation criterion for the power of the Model.
7. **Deploy** the configured Solution:
  - Prepare the implementation of the new Solution with a **changeover Pilot**
  - Reorganize the Enterprise, if necessary
  - Assign and train the Actors who will use the new Solution
  - Allocate, Install and configure the Computers (Servers and Work-Stations), if necessary
  - Migrate the Data with the help of prepared Tools, if necessary
8. **Operate** the Solution: we are no longer in the Transformation stage
  - The Actors can use the "Actions" offered (Processes and Functions)
  - A Support team helps them in case of difficulty
  - Exceptions are handled
  - The Solution is optimized
  - And when the Solution is completely generalized, we evaluate the Value indicators defined at the start to check that the Goal has been properly met.

### 4.3 Engineering improvements to enhance Agility

In the following section of this document we will develop a series of Engineering improvement proposals which can strongly boost Agility.

- Strive for a **single Model** shared by Business and IT: it will help avoid translation errors from the Business Model to the IT Model
- Define a rigorous **Business language** to facilitate communication between Business and IT
- Reuse all forms of **Components** to diminish the workload
- Model the Core Business **before** the Organization
- Apply, whenever possible, the **Cooperative** Approach rather than the Contractual Approach.

To apply these recommendations, Transformation **Roles** must evolve and **Organizations** must adapt; all of which presupposes **training** the Actors concerned.

## 5 A single model shared by Business and IT

Let's take an example: if an Enterprise wants to make Productivity gains of 10%, it must transform a certain number of its Operational Processes. Thus, it has to carry out 3 actions:

1. Define the **Problem** "Grow Productivity by 10% within a given perimeter"
2. Build the **Business Solution** which consists of redefining certain Operational Processes.
3. Translate the Business Solution into an **IT Solution** if IT is used in the Building of the modified Processes.

Function 2 "Build the Business Solution" can be carried out by "hand" (using Word or PowerPoint) or with the help of Modeling tools which guide the Business Builder to produce a formalized Business Model which uses a **Business Meta Model** (a Meta Model defines the interrelated concepts which are used to structure the Models).

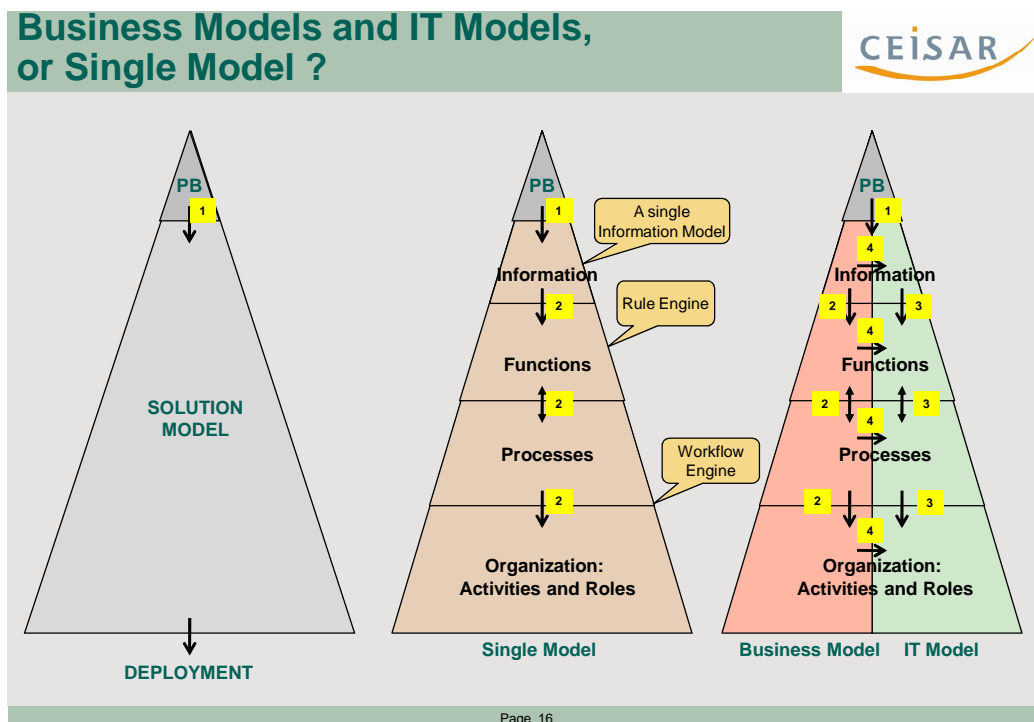
Function 3 "Translate the Business Solution into an IT Solution" is carried out by designing and programming an IT Model (the "software") which uses an **IT Meta Model**.

If Function 2 produces a formalized Business Model and if we can automatically deduce a part of the IT Model from it, that means that a part of the Business Meta Model and the IT Meta Model **overlap**.

We then use a **single Meta Model** rather than 2 separate Meta Models for the Business part and the IT part, and we offer **Views** of this Model adapted to each Actor. We no longer need to translate the Business Model into an IT Model, we no longer have to check via a test plan that the 2 Models are consistent. In other words, the Business Actor defines the Single Model and this in turn automatically produces a part of the executable IT Model (the Software). This is one means of **Fluidifying the Business/IT relation** and to **accelerate** Building.

A universal Tool to transform an entire Business Model into an IT Model does not yet exist, but progress has been made on parts of the Model, to:

- Model Data and automatically generate the IT Data Model
- Model the Business Rules directly in a rule engine
- Model a Process and automatically generate the navigation and assignment
- Model the presentation and automatically generate the GUI Model
- Model the Business Intelligence deliverables and automatically generate the production of Data



Whenever we use a single Meta Model for Business and IT we must only check that:

1. The Solution responds adequately to the Problem
2. The single Model is well Built

Yet when we use different Meta Models for Business and IT we must check that:

1. The Solution responds adequately to the Problem
2. The Business Model is well Built
3. The IT Model is well Built
4. The Business Model is properly translated into its IT Model (often called “acceptance”)

---

## 6 Defining and sharing Business language

The terms which the Enterprise uses daily such as Customer, Product, Contract, Service, Partner... are rarely defined with precision: but why waste time defining the terms that everyone already knows?

Yet in practice, each term most often regroups several Entities.

For example, "Customer" means:

- he to whom one sells (for the salesperson)
- he who signs (for the company lawyer)
- he who pays (for the accountant)
- he who benefits from the Product (for the delivery department),

We could carry out the same exercise with "Offer", "Product", "Service", "Contract", "Resource", or even "Flight" for Air France or "Train" for the French Railways...

When Business defines its Problem and its Solution Model it must use precise Business terms which will help it to hone its model and to **communicate more easily** with IT.

It is also a means of **properly structuring the Solution Model** which is based first and foremost on the Data Model and to favor the emergence of **Reusable Components**.

In practice (see White Paper on "Business Entities"), one only has to rigorously define a hundred or so terms to substantially improve dialogue.

Such a definition uses the verbs "to be" for inheritance and "to have" for relations.

It is the Role of the "Foundations Business-Architect" (see further on) to create, maintain, and communicate this Glossary.

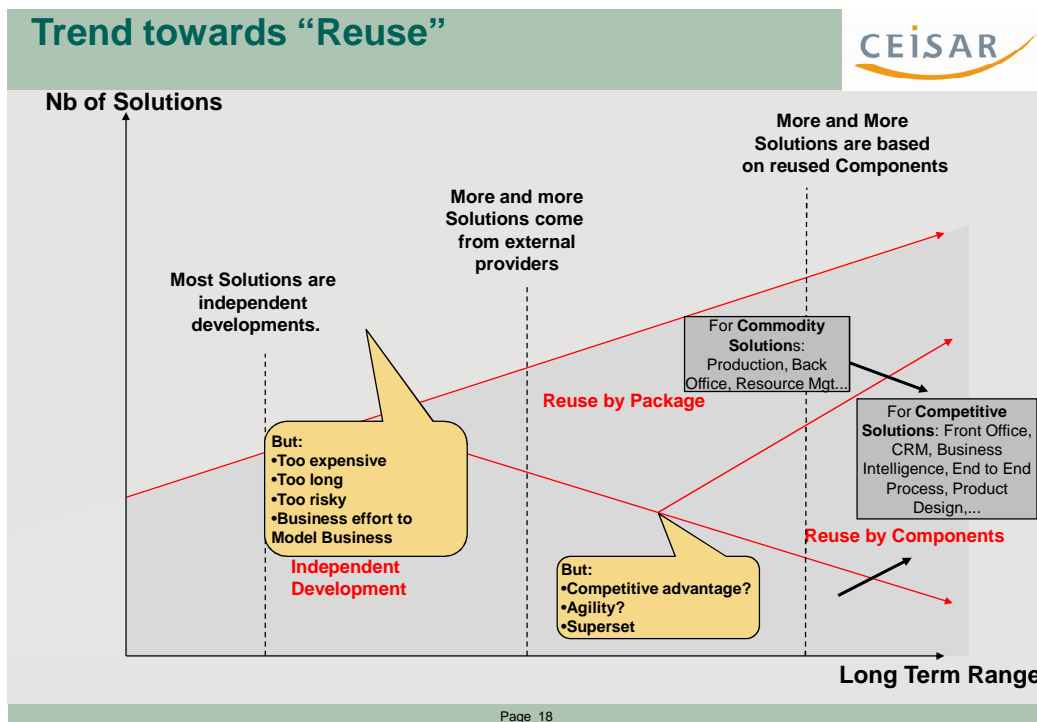
## 7 Reuse of components

There exist 2 forms of reuse: reuse by Software Package or reuse by Components.

Reuse by Software Package has had growing success in recent years for **Commodity Solutions** for which needs from one Enterprise to another are similar.

Yet a human activity only reaches maturity when it is able to reuse common components to build **Competitive Solutions**: it took 200 years for Industry to reach its current level of maturity. The design time for new automobile Models has been reduced from 5 years to 18 months via Component reuse. It will take some time before the software industry manages to do likewise. Yet hopes are high. The "SOA", "reusable component", and "Object approach" trends are all heading in this direction, and the results obtained in a certain number of Solution Models have proven that 70% reuse rates are realistic: meaning we **only need to build 30% of the Model** to satisfy a specific need. The Software Package makers are themselves carrying out this mutation: new Software Package offers are often built on a Components basis.

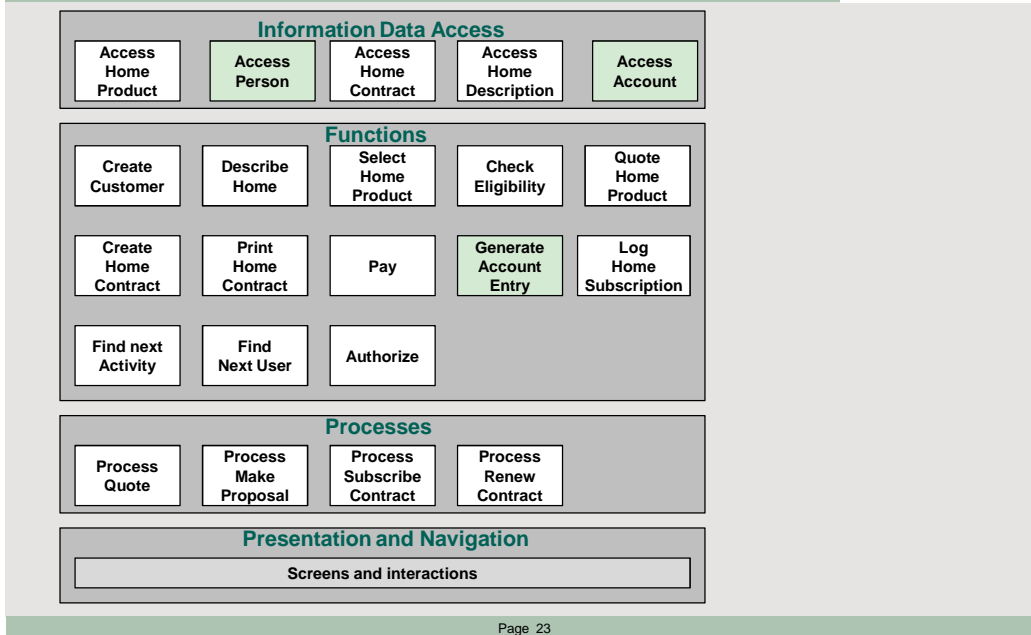
This trend also facilitates Extended Enterprise developments: a well built Model takes the relations with Partners, Suppliers and Customers into account; they become Organization-Actors in the Enterprise Architecture.



Component reuse can be very far reaching. To illustrate the potential, we took the example of an Insurance Solution broken down into "4 quarters": access to Data, Functions (Core Business or Organization), Processes, Presentation and Navigation.

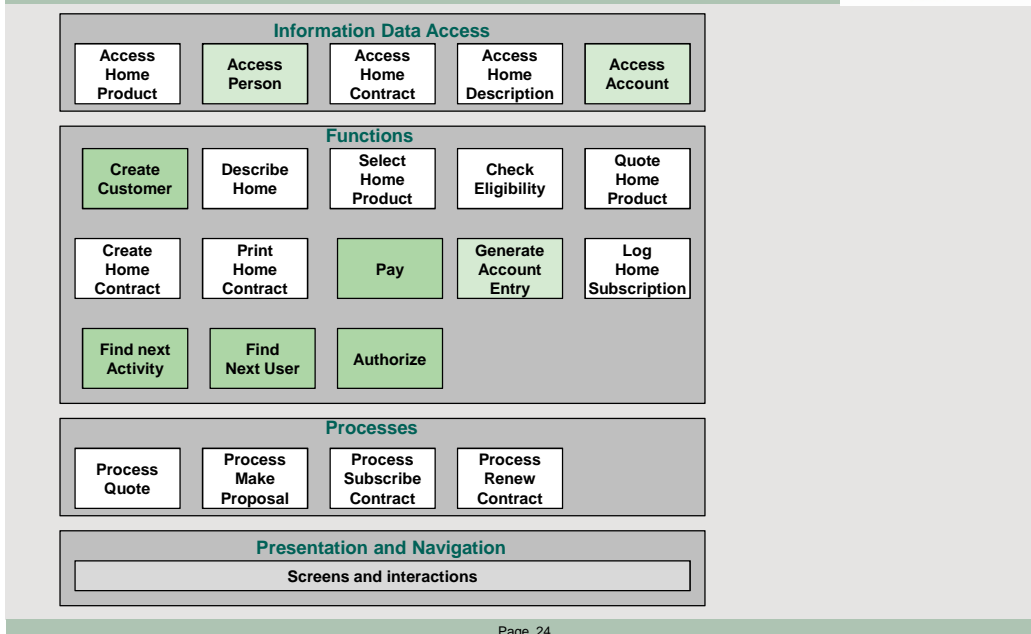
We can take advantage of 4 levels of Reusable Components which we regroup under the name of **Foundations**:

## 4.1 Information Integration (Foundation level 1)



1. Isolating **access to Information** by the shared Data access Functions enables evolutions in the Information Model and the Action Model to be made independent. It also allows the Solutions to access Information handled by other Solutions. Similarly, isolating **Flow exchanges** between internal or external Solutions contributes to overall coherence and minimizes coupling between Solutions. It is a means of **isolating each Solution** to give it evolution autonomy.

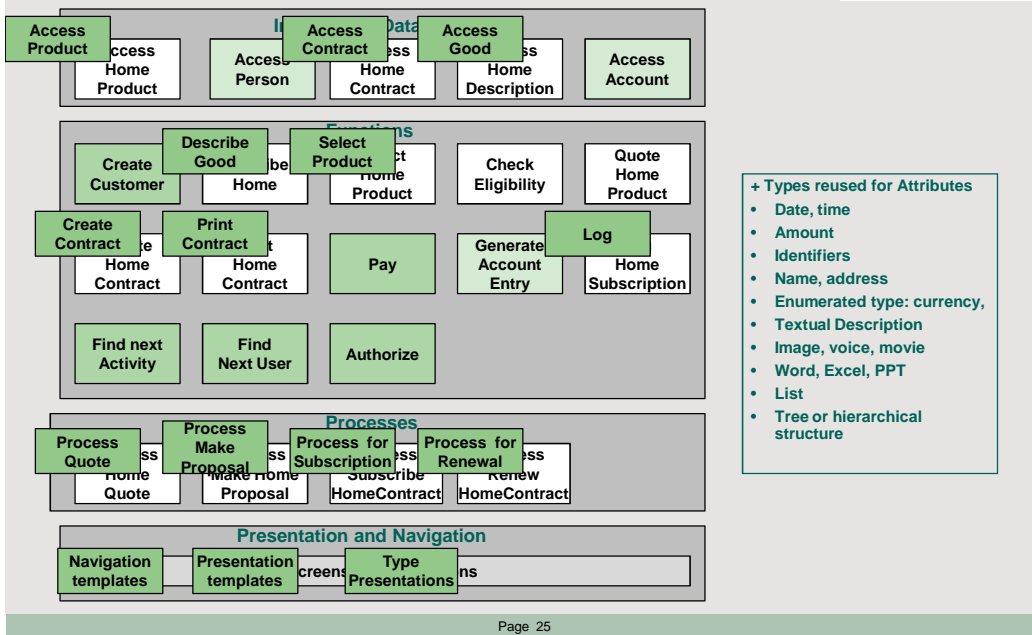
## 4.2 Reuse of Black Components (Foundation level 2)



2. Reuse of already available **Functions** to reduce what is to be built: not just the Data access and the Flow exchanges we have just described, but also Business Services (such as "Pricing") Organization Services (such as "Authorize"). This is a means of further isolating Solutions and of accelerating Building. Levels 1 and 2 are called "**Black Components**" because they behave like black boxes whose interface we know, but whose implementation we do not.

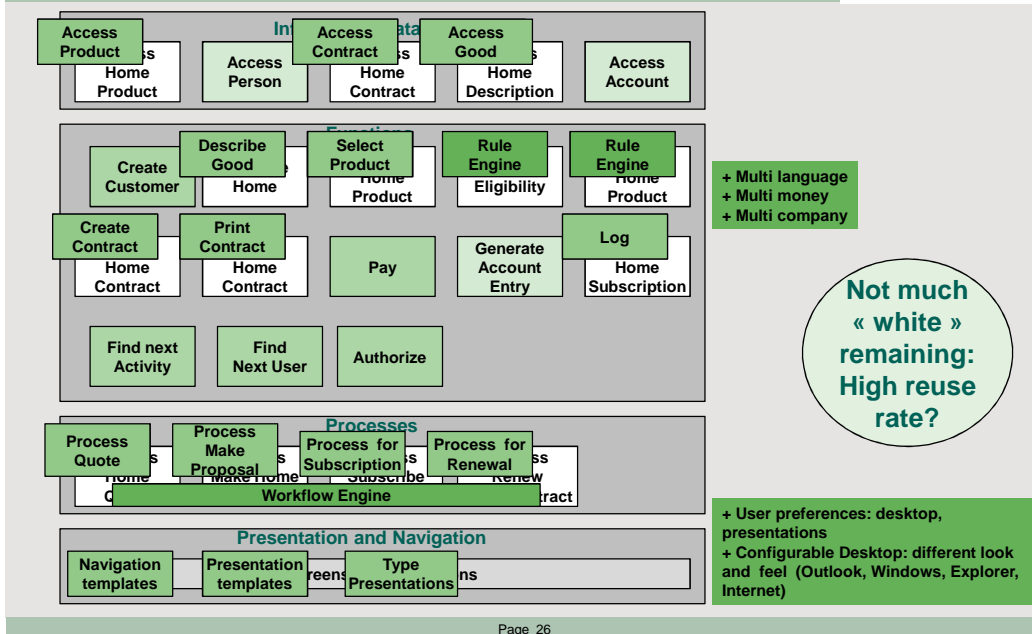


### 4.3 Reuse of White Components like inheritance, patterns, types (Foundation level 3)



- Reuse of **Patterns** or "**White Components**" such as a "subscription Pattern" which is reused by all the subscription Processes. It is a form of Reuse that differs from the 2 previous ones, as it intervenes during the Building phase and not by way of call ups in the course of Operations. It can concern the Process Models, the Function Models, the Data Models, and user Interfaces... It considerably accelerates the Building phase but requires adapted **tools**.

### 4.4 Parameters and Rule engine (Foundation level 4)



- Externalize, with the help of **parameters** and **rule engines**, those parts of the Model which change most frequently to **Configure the Model** without going through the complete Transformation Process which requires Integration, Acceptance, and onerous Production implementation. One can thus directly create new Products, change pricing, adapt.

*By applying these different levels of reuse, an international Insurance Software Package was able to pool Foundations of 6,000 classes offering 50,000 Services: the additional cost for Building a Solution adapted to each Business Line (Property and Casualty Insurance, Life, Contingency, Health...) was about **5%** of the Foundations. This level was attained thanks to favorable conditions: no legacy system to bridle innovation capacity, modern tools, experienced teams... Yet this experiment proves that we can be extremely ambitious when it comes to reuse.*

To succeed in large-scale Reuse of Components the conditions for such success must be met:

- **Isolate** in a "Foundations" team, those who build, support, and recuperate Components.
- Make sure they have acquired the **know-how** to build components: quality of interfaces, structure of small components which reuse each other – not a banal list of large components - versioning, in-depth use of Patterns, ascending compatibility, management of adapted configuration ...
- Encourage and check **reuse** in the Solution teams.

### **SOA - Remarks**

The SOA approach also deals with Components. It is characterized by:

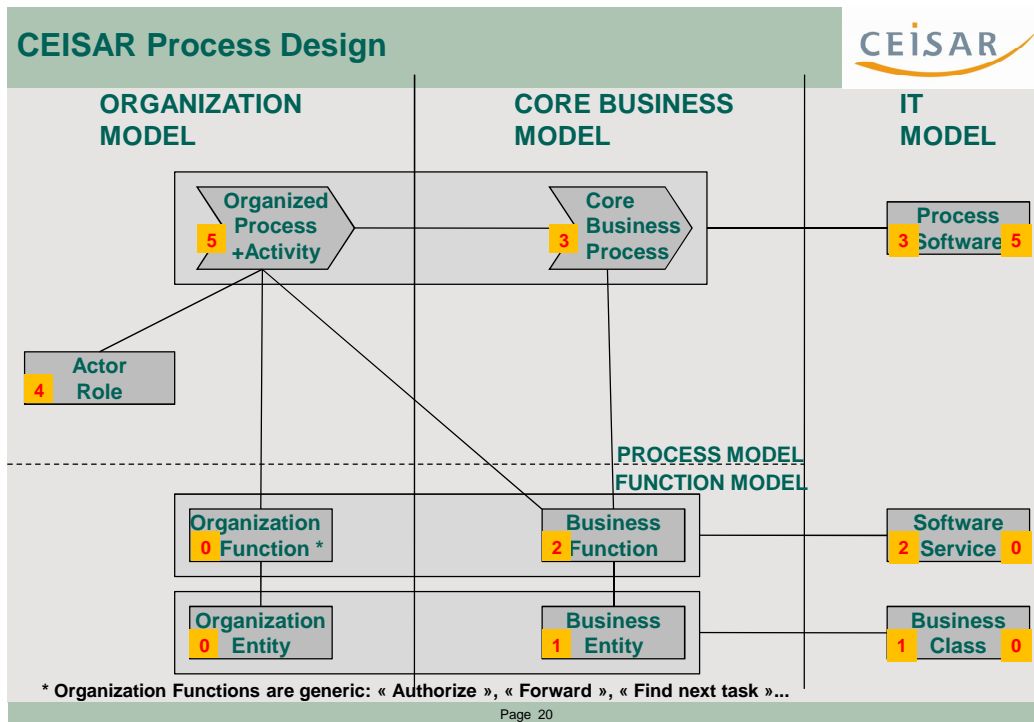
1. The use of Black **Components** (levels 1 and 2) and not White Components
2. Whose **execution** is internal or **external**; in the latter case there arise problems of security, of quality of service, of freshness of data... in short, problems of trust
3. Which need to send a response via **shared middleware** between the consumer and the Service supplier, the easiest solution being that each falls back on the market standards (today "Web Services").

## 8 Business-Process and Organized-Process

A Software Solution lasts for 20 years. Organizations are in perpetual movement. So how do we analyze Processes so that the Software that derives from them can support different successive or simultaneous Organizations with ease?

If you ask Operational Actors to describe the way they work, or the way they would like to work, they naturally describe their everyday work routine: **Actors** executing **Activities**. They will rarely speak about Processes and Data.

According to what is expressed by the Operational Actor, the Solution Builder's approach ought to be as follows:



Define the Core-Business

1. Define the **Core Business Entities** such as "Customer", "Product", "Order", "Delivery": these are the most stable elements of the Enterprise Architecture.
2. Define the life-cycle of each of these Entities to bring to light the **Core Business Functions**: "create", "check", "modify", "interrogate", "price", "debit"...
3. Define the **Core Business Processes** and break them down by reusing the Core Business Functions defined in Stage 2 and by adding new ones that can only be discovered by analyzing the Process needs.

Define the Organization

4. Define what **Roles** are attributed to what Actors or to the organizational Units which comprise them
5. Define the **Organized Processes** segmented into **Activities** (an Activity is only executed by an Actor) to assign them to the Roles defined in 4. Break down each Organized Process into Business Functions and Organization Functions (the latter belong to Foundations)

This Building approach has multiple advantages:

- The Solutions are based on the Core Business and support **changes in organization**
- We bring to light the **reusable** Entities because we analyze the Business Entities before the Processes.
- We can begin to Build the IT Solution **without waiting** for the entire Organization to be defined.
- The Solution can be easily adapted to outside partners

---

## 9 Contractual Approach or Cooperative Approach

In the **Contractual Approach** currently in practice, all the Business Modeling of the Solution is done prior to working on the IT Modeling. This is an efficient means of ensuring that the IT Solution correctly meets the Business needs, or that the "Functionalities" are Modeled before developing the Software. This is especially the case when the IT part is **outsourced** to an external team. The **Checking** Function can only be carried out once the IT part has been completed. This approach leads to separation of the Business and IT teams: the link between them has to be made via a contractual document called "detailed specifications".

In the **Cooperative Approach**, not all of the Business Vision of the Model is detailed prior to beginning the IT Modeling. The definition of Functionalities is progressively fine tuned in the course of **iterations**. This approach is preferable when specifications are uncertain and when we want to build Evolutive Solutions rather than definitive Solutions, in other words Competitive Solutions as opposed to Commodity Solutions. This approach is faster, it diminishes the "tunnel" effect, it does not oblige Business to predefine its entire Model before handing over to IT, it fluidifies the Business/IT relationship, it enables progressive checking, and it allows the bringing together of Business and IT in a single mixed team responsible for the Solution; but it can **only** be opted for **if the Solution Global Model supports later add-ons**.

The Contractual Approach is today more widespread than the Cooperative Approach. It solves **well defined** problems such as Back Office, Production, Accounting... it suits the Building of **Commodity Solutions**.

However, it is poorly adapted to new **extensible** Problems such as Marketing automation, CRM, Internet Solutions offered to customers, the handling of sales campaigns or Business Intelligence: Business Modeling in such domains is progressive; each new version of the Solution helps to outline the following version. Enterprises will have an increasing need for the Cooperative Approach for the development of **Competitive Solutions**.

### But how to Build a Solution Global Model that supports progressive add-ons?

We deploy Solution **Versions** and to Build each Version we proceed by **iteration**.

The Solution Architecture must accept successive Versions and Iterations of the Solution ([http://alistair.cockburn.us/index.php/Incremental versus iterative development](http://alistair.cockburn.us/index.php/Incremental%20versus%20iterative%20development)).

To succeed, we must respect what was previously defined:

- dissociate Core Business and **Organization**
- follow the **sequence** "Entities, Functions, Processes"
- reuse a maximum number of **Components**.

To this we can add:

- plan for all Model elements to evolve (all elements of the Model ought to integrate **Version** management)
- ask the **best** "Modelers" to devote themselves to the Building of the Solution Models or Foundations; don't overload them with pure management tasks, they are a rare resource
- have the quality of the Solution Global Model **certified** by experts.

For those who wish to pursue this further, see the CEISAR White papers on Business Entity and Process Modeling.

### A Single Model

The Cooperative Approach is more efficient if it based on a **Single Solution Model**: all modifications, whether they stem from a Business change or an IT change, update the single Model and instantly modify the views offered to each and everyone. These are what are called "Round Trip" tools (see above).

### How to stop a Version: limit the specifications by date and not by functional perimeter.

As we don't predefine all we have to do prior to doing it, we must define a rule in order to stop a Version. Perfectionism is dangerous: many projects run into difficulties because the Business Analysts are afraid of forgetting demands and do not know when to stop them. They remember the words of wisdom of the Contractual Approach "If you omit certain requests it will be long and costly to take them into account in subsequent versions..."

We must help them apply the following rule: "as it is easy to move on to a new incremental version, let us rapidly deliver a first version to offer an initial level of service to Actors..."

Of course, if it is a matter of replacing an older application, we must at least re-include the Functions offered by that former application.

The rule of thumb is to set a realistic **date limit** for the first version, which will mean that the Project Management will have to designate a leader capable of **priority sorting** the progressive demands involved so as to meet the deadline. This principle of sobriety is the key to a successful cooperative approach. The first version ought to deliver:

- The Architecture of the new Solution which must allow rapid increments
- The Functionalities already offered by the former Solution
- Some additional attractive Functionalities, as long as they are compatible with the deadline set.

### A Single Building Team for the Cooperative Approach

The Business and IT Roles are grouped within the same team which is solely responsible for the Solution. Each Solution Version is a compromise between functional wishes and technical possibilities. Outsourcing of IT development is possible. Yet it proceeds by way of small, successive contracts: an outsourcing Contract can be defined for each Version after a sufficient number of iterations have been carried out to select the requirements of the current Version.

### Is it possible to function with dispersed teams?

A Project team always functions better when its members are located under the same roof. Yet this is not always feasible. We can however implement a cooperative approach between dispersed teams if they share efficient communications tools - not just between individuals, but also between Models. The ideal solution is to offer each local team a local repository, and at Central level, to aggregate the contributions of each local team via a global repository which manages versioning, consistency, and which offers each player the views they need.

### Should IT Transformers (designers, IT developers) be kept within the Enterprise?

The Business Model must be translated into an IT Model. This translation is increasingly considered to be an outsourceable task, with low added value: thus **IT Transformers** are often given **insufficient consideration** within the Enterprise. This trend presents two dangers:

- First, if the Business Model is well built at the beginning, it is rarely updated subsequently, while the software supports successive modifications. After a number of years, the IT Model represents the true updated reality of the Enterprise Model, while the Business Model is outdated: once an Enterprise has outsourced its IT Model, it becomes increasingly difficult for the Enterprise to know precisely how it functions.
- Second, outsourcing will put a brake on the implementation of a Cooperative Approach in Enterprises: it is hard to strike a balance of compromise between functional wishes and IT possibilities if we outsource and contractualize the Business/IT relation.

It is preferable that Enterprises maintain an in-house core of recognized IT skills and that they steer the global design of the IT Model, otherwise they will one day find themselves dependent on suppliers who will have evolved the IT Solutions while the Enterprise has failed to conserve a knowledge of its own Business Model.

### The Contractual Approach and the Cooperative Approach can be used jointly

From the Cooperative Approach conserve


- Successive Versions
- Quality of the Global Model

From the Contractual Approach conserve

- Separation of Business and IT teams
- Possibility of contracting out the Building of the IT Model

Yet contracting must be carried out according to more limited perimeters defined for each Version. As one of CEISAR's sponsors has estimated, we need to have consumed 40% of the Modeling workload to obtain a 10%-accuracy evaluation: the more precise we wish to be in terms of estimation the further on we must be in terms of project advancement.

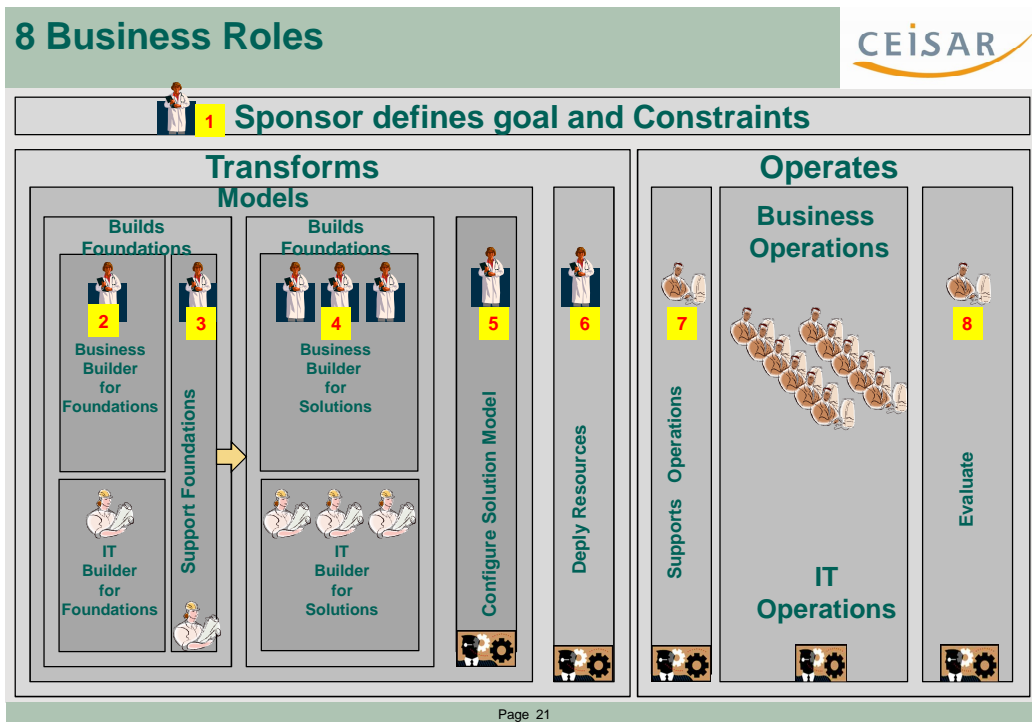
Contractual or Cooperative Approach



<p style="text-align: center; color: #008080;"><b>Why?</b></p> <p><b>Well defined needs</b> (Back Office, Production, Legal rules)</p>	<p style="text-align: center; color: #008080;"><b>Contractual Approach</b></p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p style="text-align: center; color: #008080;"><b>Conditions for Success</b></p> <ol style="list-style-type: none"> <li>1. <b>Contract</b> is important</li> <li>2. Project leader is a <b>Manager</b></li> <li>3. Business analysts <b>must</b> provide a <b>definitive</b> list of requirements. IT is in another structure</li> <li>4. Business Modeling tools are independent from IT tools</li> <li>5. Reuse of Solution Packages</li> </ol> </div>	<p style="text-align: center; color: #008080;"><b>Consequences?</b></p> <ol style="list-style-type: none"> <li>1. Contractual relation: <b>"IT do it!"</b></li> <li>2. Done when <b>all</b> requirements are satisfied</li> <li>3. Possible to Outsource all IT</li> <li>4. <b>Tunnel effect</b></li> <li>5. <b>Future Evolutions</b> could be expensive</li> <li>6. <b>No hope for Foundations</b></li> </ol>
<p style="text-align: center; color: #008080;"><b>Why?</b></p> <p><b>Evolving needs</b> (Front office, CRM, Business Intelligence, Product Design, Extended Enterprise)</p>	<p style="text-align: center; color: #008080;"><b>Cooperative Approach</b></p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p style="text-align: center; color: #008080;"><b>Conditions for Success</b></p> <ol style="list-style-type: none"> <li>1. Quality of <b>Build</b> is important to accept complementary needs</li> <li>2. Project leader is more a <b>Builder</b></li> <li>3. Business Analysts <b>must</b> provide a <b>first list</b> of requirements : they are part of a <b>mixed team</b>: Business/IT</li> <li>4. <b>Modeling Tools</b> should be shared and "Round trip"</li> <li>5. Reuse of Components</li> </ol> </div>	<p style="text-align: center; color: #008080;"><b>Consequences?</b></p> <ol style="list-style-type: none"> <li>1. Share responsibility: "Find a good Solution together"</li> <li>2. <b>Compromise</b> between Business hopes and IT possibilities</li> <li>3. V1 Done when <b>deadline</b> is up</li> <li>4. Possible to outsource for <b>each</b> successive Version</li> <li>5. <b>Fast results</b></li> <li>6. Business Analysts have <b>the right to be wrong</b></li> <li>7. <b>Foundations possible</b></li> </ol>

Page 14

# 10 The Business Roles of Transformation



As the question asked in this White Paper by the CEISAR's Sponsors also concerns the training of Project Management we **focused on the Business Roles** and not on the IT Roles. We would underline that Business and IT are intimately linked in the Transformation Process and an IT extension of this list should be defined in an upcoming White Paper.

There exist numerous Business Roles within Transformation such as:

- Information Administrator
- Analyst
- Business Foundations Architect
  - Component Architect
  - Security Architect
- Business Solutions Architect
- Business Project Manager
- Business Designer
- Solution Configurator
- Ergonomist
- Change Manager
- Methodologist
- Organizer
- Quality Manager
- Security Manager
- Business Solution Manager
- Sponsor

Rather than cite an exhaustive list of Business Roles that exist in our Sponsors' companies we have sought to simplify by defining the **global team-bound Roles**; we have defined 8 key roles:

- In addition to the Role of **Sponsor**, there exist
- 5 Roles in Transformation
  - **Business Builder for Foundations**
  - **Business Support** for Foundations serving Business Builders for Solutions
  - **Business Builder for Solutions**

- Solution **Configurator**
- **Deployer**
- 2 Roles in Operations
  - **Solution Support** serving Operational Actors
  - Operational Solution **Evaluator** ( or Process Pilot)

In each of the 8 categories, there can exist

- different **hierarchical** levels according to team size: for instance, for the Role of "Business Builder" one can define the "Business Project Manager" and the "Business Analyst"
- different specialists can **specialize** as **experts** in a part of the Model: ergonomist, organizer, security expert.

We will now propose a definition of the Role of each team. It is up to each individual Enterprise to decide if it wishes to detail several Actor Roles within each of these teams. The only recommendation we would make is not to over multiply Actor Roles so as not to multiply management and interaction tasks: if we have the choice, it is preferable to have **a few high-level Actors** - and thus polyvalent – rather than **numerous specialist Actors**.

- The **Sponsor** defines the Problem: Goal, Value and Constraints. It is the sponsor who decides, who pays and who checks that the solution correctly provides the value defined in the Problem.
- **The Business Builders for Foundations** define the **overall Solution plan** (we talk of Urbanists), they design the Business Components **common** to all the Solutions: definition of Business Entities, Process Patterns (such as the Subscription Process Pattern which can serve as a pattern for all subscription Processes), Common Business Functions (such as pricing), Common Organization Functions (like security check or ergonomic norms). They also **define the Approach**.
- **The Business Support for Foundations**: they train, help and monitor the Business Builders for Solutions, to ensure that they are properly reusing the Business Foundations and applying the Transformation Approach.
- **The Business Builders for Solutions** model the Solution (Global Business Model, Business Information Model, Business Function Model, Business Process Model, Organization Model), test the IT Solution Model that derives from it and define the Interface Model. The person in charge of the Solution Project Team is called the "**Solution Project Manager**".
- The **Business Configurators** use parametering, rule engines and workflow engines to configure the Model (pricing, control and commissioning rules...)
- The **Deployers** lead the Changeover: they reorganize Operations, plan for computer installation, train the Operational Actors, and help with the Data Migration.
- **Business Support for Operations** (or "business hot line") to help Operational Actors whenever they encounter difficulties.
- The **Solution Evaluators** (or Process Pilots) observe Operations and derive requests for improvement in subsequent versions.

### Focus on the Role of "Business Builders for Foundations"

We insist upon this Role because it is rarely identified within Enterprises, while the responsibilities we have just described are increasingly important in Organizations which are constantly seeking greater coherence, more synergies and economies of scale.

The reason is that there simply does not exist a transversal Unit devoted to Business: General Management is far from such concerns and the Business Lines Managers who pursue the efficiency of their own Business Lines do not have a mission for the common good.

That is why these responsibilities are usually handled by the IT Department: it does not have the Business expertise, but it better understands the challenges of synergy, of pooling and sharing, and of reuse because it oversees the Solutions IT Modeling for **all** Business Lines.

It is imperative to explain the importance of these concerns to General Management to obtain devoted Business quality resources within the IT Department or elsewhere.



Their essential role is to define the Business Solutions mapping, to identify the **reusable** Business Components for the various Solution Models and to contribute to the definition of the Transformation Processes (or "Approaches" or "Methodology") that the Solution-Projects must respect.

The Business Builder for Foundations:

- Defines the Solution mapping in its Business aspects
  - **Classification of Processes and Functions**
  - Solutions map and **flows** between Solutions
  - Use of shared **Data Repositories**
- Defines the **Entities** reusable in different Solutions
  - **Business Entities** such as Product, Actors, Contract, Account, etc.
  - **Organization Entities** such as Unit, Role, Profile, Right, Task Basket,....
  - **Specifies for each Entity:** textual Definition, Relations with other entities, Patterns or Entity inheritance, identifiers, versioning, Attributes, Types
- Defines the reusable **Functions:**
  - **Core Business Functions** such as Pricing, eligibility Control, Customer information Search...
  - **Organization Functions** such as Authorize, Assign Activities to Actors, Manage the tasks of each Actor, Journalize Actions, Manage the user Interface
  - **Specifies for each Function:** what its purpose is, what information it needs to receive in order to execute

Remark: the Business Builder for Foundations is not really involved in reusable technical Functions such as peripheral interfaces (DMS, printing, card readers...) or network interfaces

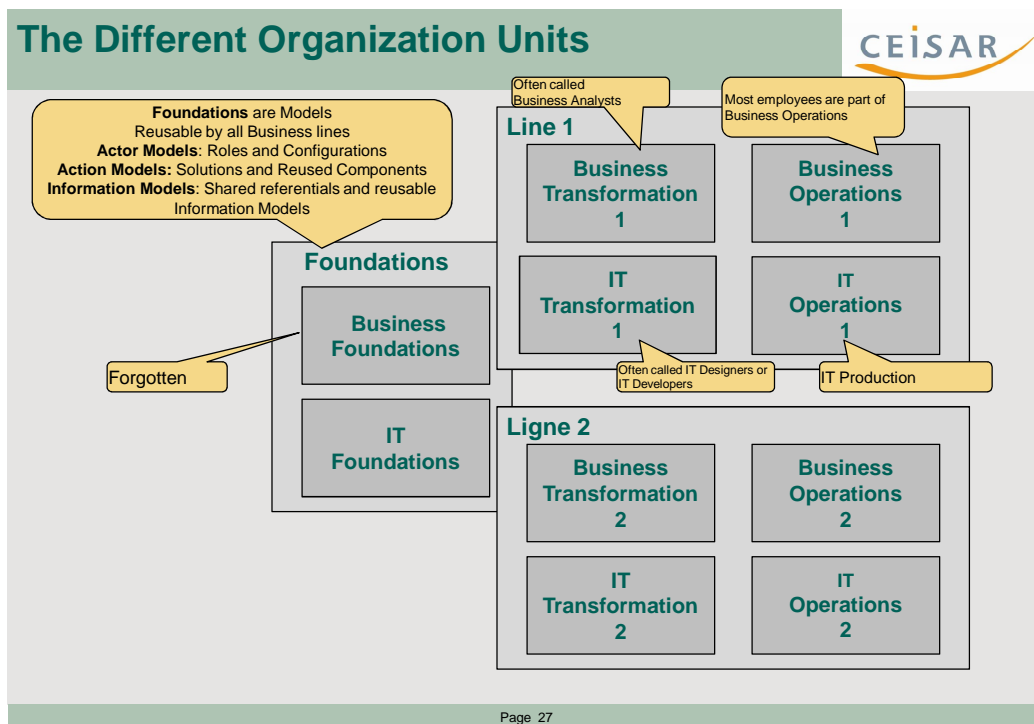
- Defines the reusable **Business Process Patterns** such as the subscription Process Model which can serve as a pattern for all subscription Processes
- Defines the **user interface** norms: presentation, navigation, printing norms
- Defines the **Transformation Processes** which the Business Solution Analysts must respect
- Chooses the **tools** to automate certain Transformation Process Functions.

# 11 Organization

Once the Roles have been defined, we can choose an Organization.

Various dimensions must be distinguished:

- Business and IT
- Operation and Transformation
- Business Line Solution and Foundations



In order to enhance agility, we must:

- **Isolate the Foundations:** join Business and IT skills within the same team to produce reusable Components and an adapted Approach. They must be given the power to have Component reuse respected.
- **Separate Operations and Transformations** because present concerns always take priority over future concerns. This separation can take place within each Business Line or globally for the Enterprise.
- Each Transformation team must **bring together Business and IT skills**, even if adopting a contractual approach, because compromises are thus worked out at the closest level to the Projects: apply the principle of subsidiarity.
- The Enterprise must **control its Global Model** and only outsource detailed Models: this is the key to its evolutivity.

## "Contracts"

According to the form of Organization chosen, there can exist different kinds of contracts between Actors:

- A contract between the Sponsor and the global Project Manager which is the description of the **Problem**
- A contract between the Business Project Manager and the IT Project Manager which can be the **Global** or **Detailed Business Model** according to whether or not the IT team includes Business Builder assistants .
- A contract for IT outsourcing which can be the **Business Model** or the **Global IT Model** according to whether or not we are outsourcing the overall IT design.

## The qualities of a Project Solution Leader: Builder and Manager

- A project leader who is simply a **Manager** without **Modeling** talents won't have the common sense reflexes of someone who knows what is realistic, what is easily implementable, or when the Solution Architecture should be updated. To offset this lack of judgment or experience, he will tend to protect himself through an excess of checking procedures, reporting, or meetings to obtain consensus.
- A project leader who is simply a **Modeler** without **Management** talents will be faced with other well identified problems: for instance, not knowing when to stop the flow of demands, or neglect of reporting or documentation aspects.

If the project leader has both Business and IT expertise, it is a perfect but rare profile.

If they don't have such expertise, they must surround themselves with experts who provide it.

---

## 12 Training of Business Transformers

If we want that Business Builders genuinely take Transformation on board and collaborate efficiently with IT, they have to be trained in Transformation. They must have assimilated the **Approach**, the Transformation **Tools**, the **Foundations** and the Architecture of the **Model in place** which they have to Transform.

The prerequisite to all training is thus to define the Approach, the Tools and the Foundations.

If it involves the Approach, Tools and Foundations already **in place**, this training can be carried out by the in-house teams who already use them.

If the Enterprise wishes to **improve** its Approach, Tools and Foundations, then it is necessary to call on outside experts who can bring new elements to the teams already in place: concrete examples of Enterprise Projects which succeeded with different Approaches, Tools and Foundations.

The Training Modules which Business Transformers ought to follow are as follows:

- Understanding what **Enterprise Architecture** is: alignment with Strategy, the break down into Operations and Transformations, the break down into Real World and Model, the quest for Synergies through Model reuse and Sharing of Resources
- Understanding what an **Enterprise Model** is: Data Model, Action Model (Processes and Functions) and Actor Model (Human Actors and computers)
  - Knowing how to execute a Project: the training ought to be based on the Transformation Process in use within the Enterprise. Upon describing this Process we must highlight the **break down into 8 Engineering Functions** (Understanding the Context, Defining the Problem, Building the Solution, Checking the Solution, Adapting the Solution, Configuring the Solution, Deploying the Solution, Operating the Solution).
  - Understanding the **Cooperative Approach** and the **Contractual Approach** and assimilation of the associated Management Functions:
    - splitting into Phases
    - the Role of Transformation Actors and the allocation of Actors to Phases
    - how to iterate
    - how to evaluate the workload and cost of a Transformation
    - the Decision-making and Checking Process ("Governance")
    - the Planning Process and the follow-up Process
    - the handling of Changes in the course of the project
    - the Documentation to be produced
- **Personal Skills**: preparing the Business Transformers to separate structure from detail, and not to rely on an "unquestioning" respect for the Transformation Process to make their project a success
- **Knowledge of Foundations**: first, we must get the idea across that it is possible to Build Solutions for highly **specific** needs with the help of **common** Foundations. This is the underlying principle of reuse, but is not straightforward. Once this principle has been acquired, the variety of Components that one can reuse must be demonstrated. If the Foundations do not yet exist, the potentialities must be highlighted: if they do exist, they must be described.
- For each Engineering Function it must be made clear what must be handled by the Business Team and what by the IT Team:
  - Define the **Problem**,
  - Build the **Business Entities**: create, modify or reuse,
  - Build the **Business Functions**
  - Build the **Business Processes**,
  - Build the **Organization Entities**,
  - Build the **Organization Functions**,
  - Build the **Organization Processes**,
  - Build the **Roles**,
  - Build the **Training Processes**,
  - Build the data **Migration Processes**,
  - **Check and Test**

**By way of conclusion:**

The appetite for Competitive Solutions is set to grow considerably, fuelled by the financial and economic crisis. Competitive Solutions require a new Cooperative Approach which implies far-reaching cultural change whose difficulty should not be underestimated. Enterprises must waste no time in taking on this new challenge: General Management ought to be informed and pilot projects launched to demonstrate the quality of this approach before generalizing it.