Le projet **DELIRE**

Développement par **E**quipe de **L**ivrables **I**nformatiques et **R**éalisation **E**ncadrée

GP3 – De l'architecture fonctionnelle à l'architecture logique

Abstraction, Conceptualisation, Modélisation

Il ne viendrait à l'idée de personne de construire une maison en ne commençant pas par les fondations.
Et pourtant, en informatique, combien est-il classique de voir des ingénieurs se précipiter pour coder dès qu'on leur a soumis un problème. Le résultat étant souvent à la hauteur du peu de temps passé à analyser le problème et concevoir la solution.

Abstraction, Conceptualisation, et Modélisation sont les fondations de tout projet. Que ce projet soit informatique, ou qu'il s'agisse de la conception d'un pont, d'un avion, ou d'un stylo.
Mais c'est une démarche peu naturelle, tellement une première solution semble toujours évidente de prime abord.

Le temps, a priori perdu en début de projet à identifier le besoin, à explorer les solutions alternatives, à manager les risques majeurs du projet ou à définir avec précision la responsabilité de chaque acteur, se révèle à terme un gain précieux. En phase de réalisation, chaque composant s'élabore sans difficulté et trouve naturellement et en temps sa place dans l'assemblage final, avec un degré de qualité satisfaisant.

Avant de vous laisser poser les mains sur un clavier dans le projet DELIRE, je vous ai demandé de spécifier le besoin, d'identifier le marché. Rien ne sert de faire un beau logiciel si on ne garde pas à l'esprit ces deux questions :

1. A qui va-t-il servir ?
2. Quelle sera sa valeur ajoutée pour le client ?

Mais, une fois le besoin identifié, j'ai insisté sur le travail sur les SFG (les Spécifications Fonctionnelles Générales) et l'architecture logique du produit.

Expression des besoins

Un projet c'est la conception d'un produit destiné à un client. Il est donc nécessaire que le client ait préalablement exprimé ce qu'il attend, quels sont ses besoins.

Le premier acte d'un projet est l'expression du besoin : que cherche-t-on à obtenir. Ainsi, si vous voulez faire construire une maison, il est nécessaire de spécifier si vous êtes plutôt teuf avec réception de copains ou cocooning désirant une grande nichée, si vous souhaitez faire un investissement ou garder cette maison jusqu'à votre retraite. Bref quels avantages en attendez-vous.

Le second acte est l'étude de faisabilité :

1. Compte tenu de vos besoins, de votre budget et de l'urgence de la résolution du problème, le projet est-il réaliste ?
2. Existe-t-il une solution technique à votre demande.

Le cahier des charges doit être exhaustif de tout ce qu'attend le client :

1. Les fonctionnalités attendues
2. Les contraintes à respecter : contraintes techniques, respect d'une norme ou d'une législation, sous-traitant à employer,
3. Les objectifs de qualité à atteindre : performance, disponibilité….
4. Le budget qu'il est prêt à y consacrer et le plan de financement
5. La date à laquelle il attend la livraison du produit final

Ces attentes s'appellent les exigences. Et toute exigence qui ne sera pas exprimée sera considérée comme de la libre appréciation du fournisseur

Attention, pour qu'un cahier des charges soit crédible, il faut que chaque demande soit chiffrée. Les expressions générales du type : « un logiciel performant et fiable, disponible le plus tôt possible » ne définissent pas un cadre de négociation.
Une exigence est soit chiffrée (décollage en moins de 1200 mètres) soit booléenne (pas d'utilisation de plomb).

Le cahier des charges doit également clairement exprimer ce qui est obligatoire de ce qui peut faire l'objet de discussion. Ainsi, dans la charge « 5 chambres de 12 à 15 m^2 », le nombre de chambre n'est probablement pas négociable, mais la taille de chaque chambre peut être discutée.
De même dans ses exigences le client doit définir ses priorités : le plus important est-il le nombre de chambres ou la cuisine et la salle à manger séparées ?

La décomposition en architecture fonctionnelle

Une fois le cahier des charges établi par le client, il est nécessaire de formaliser la réponse par le fournisseur.

Sachez que dans une approche de type appel d'offres, chaque fournisseur potentiel va proposer une solution qui ne répondra généralement pas à 100% au cahier des charges :

1. Incohérence entre Délai, Coût et Qualité demandés
2. Non maîtrise d'une solution technique
3. Savoir-faire spécifique
4. Proposition d'une déviation, sans conséquence majeure sur l'offre de services mais réduisant fortement le délai ou le coût.

Bien entendu, chaque fournisseur va construire une proposition qui masque ses points faibles et met en évidence ses facteurs différentiateurs.

Lorsqu'il va construire sa réponse, le fournisseur doit prendre en compte plusieurs aspects au regard de la dynamique attendue de son produit.

1. Une fonctionnalité doit-elle être générique ou spécifique
2. Un projet doit toujours prendre en compte les évolutions futures du produit.

Une fonctionnalité doit-elle être <u>générique ou spécifique</u>
En règle générale on considère que le coût d'une fonction générique est deux fois supérieur à une fonction spécifique. Il est donc nécessaire de bien analyser la demande pour voir si plusieurs demandes ne pourraient pas être fédérées au sein d'un même service.
D'autre part, vous n'apporterez pas la même réponse selon que vous êtes éditeur de logiciel (en ce cas, votre intérêt est de vous constituer une bibliothèque de composants génériques) ou société de services à façon.
De même vous devez anticiper les évolutions de votre produit : cette demande aujourd'hui spécifique ne va-t-elle pas se généraliser ?
Enfin le budget peut vous conduire à faire au plus juste. Une solution peut être de définir un service qui a une interface générique et une implémentation spécifique. Ainsi, la conversion du service en un service générique se fera localement, sans impact sur l'architecture globale future du produit.

Un projet doit toujours prendre en compte les <u>évolutions futures du produit.</u>
Défaire un travail coute toujours de l'argent, et peut se révéler très cher. Ainsi, même si vous n'installez la salle de bain que dans deux ans, vous aurez prévu l'arrivée et l'évacuation des eaux.
De même dans votre travail d'architecture et de définition de votre infrastructure devrez-vous avoir prévu les évolutions futures de votre produit. Et faire un travail qui dépasse la demande de votre client. Ainsi peut-il être judicieux de prévoir une colonne technique pour pouvoir à terme transformer une chambre en salle de bain. Mais comme toute peine mérite salaire, n'oubliez pas de mettre ce plus en évidence dans votre proposition.
Dans le cas de DELIRE, si vous choisissez de réaliser en 2 ou 3 passes votre coding, vous aurez à prendre en compte cet aspect du projet.

L'ensemble des propositions du fournisseur s'appelle les spécifications fonctionnelles générales. On y retrouve de façon exhaustive :

1. Les fonctionnalités attendues
2. Les contraintes prises en compte
3. Les objectifs de qualité à atteindre : performance, disponibilité….
4. Le coût du produit
5. La date de livraison au plus tard du produit final

Chaque spécification fonctionnelle générale est soit chiffrée soit booléenne.
Attention : dès lors que le contrat va être signé chaque spécification doit être tenue. Sachez donc être suffisamment clair dans vos propositions pour plaire au client sans être trop explicite. Ainsi, si dans vos spécifications vous proposez un User Interface, sachez vous donner des libertés de manœuvre pour en garder l'esprit mais ne pas vous contraindre sur la lettre.

Le choix des spécifications fonctionnelles générales optimales est, avec le travail d'architecture, le domaine où il faut au maximum solliciter le génie de l'ingénieur. Là aussi l'expérience se construit avec le temps, mais quelques règles « de bon sens » peuvent vous aider à éviter certaines déconvenues.

1. Appuyez-vous sur votre savoir-faire.
    a. Evitez de vous lancer dans des pistes que vous ne maîtrisez pas.
    b. Ceci étant posé, un projet est nécessairement une prise de risque, et l'absence d'innovation conduirait rapidement à l'obsolescence.
    c. On considère que 10% d'innovations sont un chiffre raisonnable à ne pas dépasser
2. Le meilleur rapport qualité prix pour le client
    a. Les développeurs adorent se faire plaisir, soit en se lançant un défi, soit au contraire en évitant tout risque
    b. C'est souvent la simplicité qui est bonne conseillère
    c. Entre deux choix, ce qui doit vous guider est l'intérêt du client
3. Un système cohérent
    a. Il faut que la finalité du produit apparaisse à la simple lecture des spécifications fonctionnelles générales
    b. Les protubérances ou au contraire les manques sont la marque d'une analyse fonctionnelle insuffisante, ou au contraire d'un cahier des charges incomplet
    c. Pensez générique
4. Un système construit pour grandir
    a. Votre produit sera souvent appelé à se développer et à grandir
    b. Ne faites pas de choix qui vous contraindront pour le futur.

Les spécifications fonctionnelles générales vont ensuite être décomposées, au travers du travail d'architecture, jusqu'à obtenir les spécifications détaillées de chaque composant élémentaire.
Les deux mauvaises découvertes en fin de conception et de réalisation sont :

1. Il me manque ça :
    - Ce petit service stupide mais qui peut remettre en cause toute votre conception.
    - Je me souviens d'un rapport de la Cour des Comptes qui mettait en lumière la conception d'une piscine pour laquelle, le jour de son

inauguration, on à découvert qu'elle n'était pas relié au réseau de distribution d'eau.

2. A quoi ça sert : ce développement certainement brillantissime qui vous a couté tellement, mais qui finalement n'apporte rien dans l'ensemble.

C'est pourquoi dans la démarche systémique on a défini la notion d'exigences. Et lorsqu'on décompose les composants en sous composants, chaque fonction proposée doit répondre à une exigence. C'est la notion de traçabilité des exigences : répondre au pourquoi on fait quelque chose. Durant cette décomposition, qui est le travail d'architecture, on fait apparaitre de nouvelles contraintes techniques. Nous reviendrons sur ce point dans le chapitre Architecture.

Ok Antoine, mais finalement comment procède-t-on ?
Identifiez l'ensemble des spécifications nécessaires pour répondre à la demande du produit
Pour ce faire identifiez les grandes macro-spécifications puis décomposez les jusqu'à arriver aux spécifications élémentaires

Pour chacune des spécifications, définissez sa priorité (prioritaire et non prioritaire) et son coût (coûteuse et non coûteuse). Vous allez ainsi décomposer les fonctionnalités en 4 groupes :

1. Fonctionnalité prioritaire et non coûteuse..
2. Fonctionnalité prioritaire et coûteuse.
3. Fonctionnalité non prioritaire et non coûteuse.
4. Fonctionnalité non prioritaire et coûteuse.

Commençons par nous intéresser aux fonctionnalités non prioritaires. La question que vous devez vous poser est : chacune d'elle est-elle vraiment nécessaire ? La première réponse est généralement oui, mais après réflexion vous allez vous rendre compte que certaines fonctionnalités ne sont pas franchement nécessaires. Quitte à ce qu'une fonctionnalité prioritaire soit un peu étendue pour couvrir le besoin. Souvenez-vous que la première qualité d'un logiciel est la simplicité, qui passe en particulier par le nombre réduit de fonctionnalités offertes.
Ceci étant posé, il vous restera des fonctionnalités non prioritaires, certaines non couteuses, d'autres couteuses.

Ensuite tartinez comme suit vos fonctionnalités dans les 3 versions

1. Version basic : fonctionnalités prioritaires et non coûteuses. .
2. Version standard : fonctionnalités prioritaires et coûteuses, fonctionnalités non prioritaire et non coûteuses.
3. Version advanced : fonctionnalités non prioritaires et coûteuses.

Puis regardez la logique de chacune des versions :

1. A-t-elle une vraie valeur ajoutée
2. Est-elle complète, c'est-à-dire a-t-elle une cohérence intrinsèque pour adresser un marché

Vous allez devoir passer des fonctionnalités d'une version à une autre pour donner à chaque version sa valeur ajoutée et sa cohérence.

Que faire quand on tombe sur une fonctionnalité ultra-prioritaire (c'est-à-dire qui ne peut pas attendre la version standard) et coûteuse ?
Voir si on peut la décomposer en 2 fonctionnalités : l'une ultra-prioritaire et moyennement coûteuse, l'autre moins prioritaire et coûteuse.

Ainsi, vous avez identifié une fonctionnalité « administrer la base de données pour créer les organisations, créer les utilisateurs, gérer leurs droits, définir les composants de l'ouvrage et les affecter aux différentes organisations ».
Cette fonctionnalité est a priori une nécessité pour la version basique : ultra-prioritaire. Mais elle est également coûteuse.

Vous allez donc la décomposer en :

1. Fonctionnalité basique : gérer par script
    a. Livrer un script pour créer une organisation
    b. Livrer un script pour créer un utilisateur et ses droits
    c. Livrer un script pour créer un composant
    d. Livrer un script pour affecter un composant à une organisation
2. Fonctionnalité advanced : offrir les fonctions d'administration en interactif.

Super Antoine, mais ton approche présuppose que nous sachions si une fonctionnalité est coûteuse ou non. Comment le savoir ?

1. Première réponse : fiez-vous à votre instinct. Vous commencez à avoir suffisamment d'expérience en informatique.
2. Seconde réponse : vous saurez exactement le coût de chaque fonctionnalité lorsque vous aurez défini votre architecture logique.

Une fois une première mouture de l'architecture fonctionnelle définie, on attaque l'architecture logique.

L'architecture logique

Dans PSS7 – Architecture logique, j'ai décrit une méthodologie pour travailler sur l'architecture de DELIRE.

Ce qu'il est important de comprendre est la notion de cohérence et de couplage des composants.

Commençons par la cohérence. Votre composant doit être cohérent en tant que produit, en tant que process et en tant que ressource.

Votre composant est cohérent en tant que produit. C'est-à-dire qu'il a sa logique intrinsèque, qu'il est facile de définir son rôle dans l'architecture du produit.
Si par contre, il apparait que le composant est un fourretout, reprenez votre copie et splittez le en autant de composants qu'il sera nécessaire pour obtenir des composants cohérents en tant que produit.

Votre composant est cohérent en tant que process. C'est-à-dire que son développement (incluant l'écriture et la réalisation des tests unitaires) pour une version donnée apparaitra sous forme d'une tache dans le planning.
(Par contre, ne posera pas de problème le fait que sa version finale sera obtenue par 3 itérations : 1ère version basic, 2nde version standard, 3ème version advanced).
Se lancer dans la réalisation d'un composant demande un temps de montée en puissance : tartiner la réalisation du composant en le segmentant dans le planning n'est pas une approche raisonnable.

Votre composant est cohérent en tant que ressource : le réaliser nécessite un savoir-faire bien précis, de façon à pouvoir le confier à une même personne. Si plusieurs (N) savoir-faire sont nécessaires, c'est que le composant peut être décomposé en N composants cohérents.

Que dire du couplage
Si un composant fait appel à un nombre trop important de composants pour pouvoir implémenter ses services, ou si les services échangés entre 2 composants sont trop nombreux, cela révèle un problème d'architecture.
Trouver la bonne architecture est souvent long et pénible, mais en général une fois trouvée elle apparait comme lumineuse et on s'étonne de ne pas l'avoir trouvée plus tôt. Mon critère empirique : l'architecture doit pouvoir se représenter de façon simple.

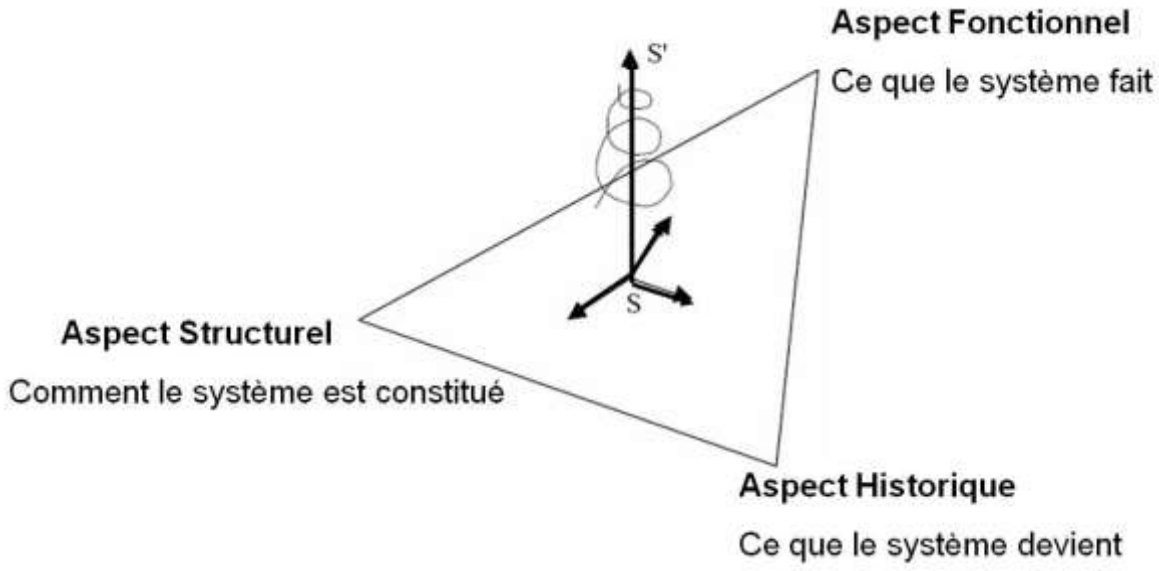
Enfin, anticipez les évolutions de votre composant, pour ne pas avoir à casser votre architecture à la première évolution. Jeune ingénieur chez Dassault Aviation, j'étais responsable du développement du programme d'aéroélasticité (le couplage entre l'Aérodynamique et la Résistance des Matériaux). J'avais défini la structure e ma base de données, qui conditionnait toute l'architecture du produit. Un matin, j'ai perçu que quelque chose n'allait pas. Ce n'était initialement qu'une intuition, mais elle m'empêchait de pouvoir travailler sereinement J'ai mis plusieurs jours à trouver le défaut de la structure de la base de données : non évolutif. Une fois identifié, il ne m'a pas fallu 10 minutes pour prendre la décision de tout casser et de tout recommencer. Au grand dam de mon chef qui a poussé des cris d'orfraie (et pour cause, le programme était financé par un contrat DRET) mais qui m'a ensuite

soutenu quand je lui ai expliqué toute la démarche qui m'avait conduit à cette décision.

Bonne nouvelle, vous avez maintenant pour chaque fonctionnalité définie dans l'architecture fonctionnelle, et plus précisément pour celle de la version basic, une bonne idée de ce qu'il va falloir développer. Vous pouvez donc identifier si elle est coûteuse ou non. Et la solution préalablement identifiée va peut-être devoir évoluer.

Architecture fonctionnelle et architecture logique cheminent de concert.

Au bout de quelques expériences en développement de gros logiciels, vous finirez par comprendre que chaque composant est un mini système, et que la démarche pour le concevoir est la démarche classique de la systémique



.

Ne l'oubliez pas : l'architecture, c'est LE savoir-faire par excellence de l'ingénieur.

L'ancienne organisation des entreprises aéronautique : la WBS

A la fin du 20ème siècle, seules les entreprises structurées étaient capable de sont capables de concevoir des produits complexes

Ceci était vrai dans l'industrie

1. Aéronautique: Airbus, Boeing
2. Automobile: General Motor, Toyota
3. TurboPropulseur: Rolls Royce, Engine Alliance
4. Ferroviaire: Alsthom, Bombardier

Mais également dans le monde informatique

1. Processeur: Intel, AMD
2. Computer: Dell, HP/Compaq
3. Software: Microsoft, IBM

Cette structuration de l'organisation, appelée WBS (Work Breakdow Structure) représentait un savoir-faire d'exception pour l'entreprise

Ce savoir-faire avait été construit pendant de longues années (Le temps étant bien sur relatif à l'histoire de l'industrie) ou par acquisition (par concentration d'entreprises, en profitant des erreurs de stratégies des concurrents)

C'est pourquoi, dans un monde concurrentiel, le cout d'entrée était prohibitif et l'arrivée de petits nouveaux rarissime.
On peut certes m'opposer Airbus comme cas d'école, mais dans les faits Airbus s'est construit sur les composants de l'industrie aéronautique européenne (Aérospatiale, DASA, BAE, CASA).

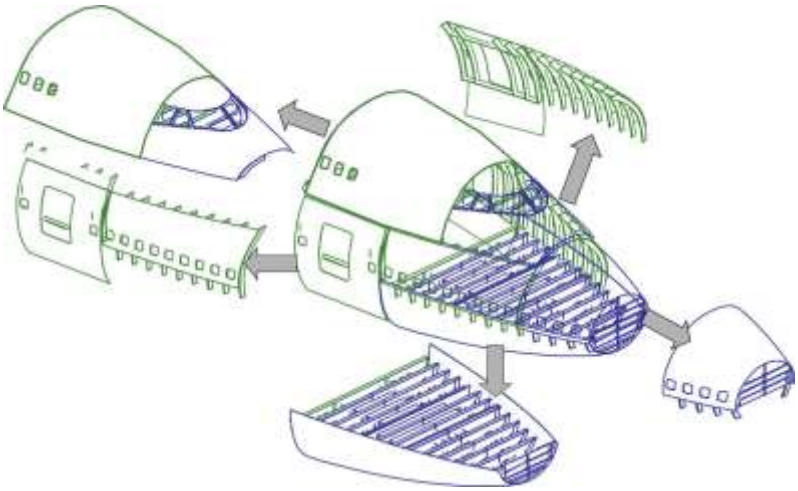
La WBS, structure de base autour de laquelle était organisée l'entreprise, définissait tout à la fois une macroarchitecture de produit, une macroarchitecture de process et une macroarchitecture de ressources.

1. D'un projet sur l'autre dans l'entreprise, on retrouvait globalement le même découpage.
2. Plus étonnant, dans deux projets de deux entreprises du même secteur, on retrouvait également le même découpage. Les entreprises aéronautiques américaines avaient normalisé la nomenclature de la WBS, ce qui permettait aux ingénieurs de passer facilement d'un programme chez Boeing à un programme chez McDonnell Douglas puis à un programme chez Lockheed

1. Fuselage
    a. Fuselage avant
        i. Section 41
            1. Poste de pilotage
            2. Accroche train avant
            3. Bouclier et radar
        ii. Section 43
    b. Fuselage central
    c. Fuselage arrière
2. Voilure
3. Moteurs

4. Gouvernes
    a. Ailerons
    b. Empennage horizontal
    c. Dérive
5. Trains d'atterrissage
    a. Train avant
    b. Trains principaux
6. Electricité
    a. Signalisation
    b. Gouvernes
    c. Passagers

…



Plusieurs décompositions intervenaient dans la définition de la WBS

1. Par zone: Section 41, Section 43
2. Par discipline: Electricité
3. Par sous traitance: Trains d'atterrissage, Moteurs
4. Connectique…

Et d'ailleurs le critère de décomposition pouvait évoluer en fonction de la profondeur dans la WBS.

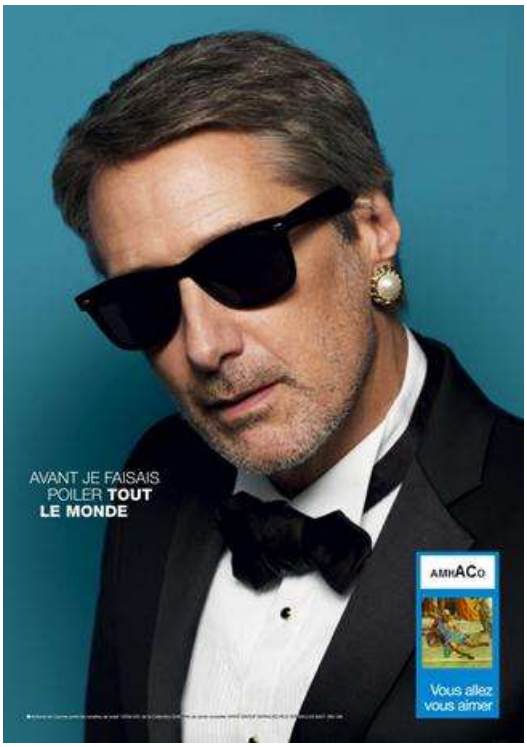
Mais le critère clé de cette décomposition était la capacité à répartir le travail.
Répartition qui pouvait être :

1. Externe
    a. Sous-traitance
    b. Co-Design
2. Interne
    a. Entre les compagnies du Groupe
    b. Entre les départements de l'Entreprise

Ce qui permettait cette grande stabilité était le fait que les programmes se déroulaient tout le temps de la même manière.

1. Le département d'aérodynamique définissait la forme extérieure
2. Le bureau d'étude définissait toutes les pièces de l'avion
3. Le bureau de calcul validait la tenue aux efforts
4. Le bureau des méthodes définissait la gamme d'usinage
5. Le SAV définissait la logistique de maintenance
6. Et tout le monde se foutrait du recyclage.

Et que l'entreprise possédait tous les savoir-faire.



Mais ça, c'était avant.

Vers une nouvelle organisation centrée sur les systèmes.

L'évolution du travail dans les entreprises s'est faite sous l'influence de plusieurs facteurs

1. Les processus généraux : globalisation, économie à la demande, financiarisation
2. Basculement vers un nouveau mode de conception : le concurrent engineering
3. Capacité des logiciels PLM
4. Capacité des réseaux, permettant le travail en architecture distribuée

Mais surtout, le nombre de savoir-faire à mobiliser et le nombre d'ingénieurs impliqués dans la conception d'un nouveau produit étaient incompatibles avec les capacités des entreprises. La mort dans l'âme, elles ont dû se résoudre à travailler en coopération avec d'autres entreprises.

Autre facteur à prendre en compte, l'informatique, l'électronique, les évolutions dans les matériaux ou l'électricité, mais également les capacités de la maquette numérique, ont fortement réduit la commonalité entre les différents programmes de l'entreprise.

Aujourd'hui, lorsqu'un nouveau programme démarre, il suit globalement le processus suivant :

1. L'entreprise responsable commence par réaliser, avec quelques cotraitants, la phase d'avant-projet pour identifier toutes les solutions possibles et retenir la meilleure
2. Puis elle choisit l'ensemble des partenaires (cotraitants, sous-traitants) qui participera au programme. Les ingénieurs des différentes entreprises se retrouvent sur un même plateau réel. Durant cette phase de preliminary design, on élabore parallèlement l'architecture du produit à terminaison, et le process que le programme va suivre
3. Enfin chaque ingénieur repart vers son entreprise mère, et le travail de detailed design se fait localement. On dit qu'on travaille alors en plateau virtuel. La base de données est répliquée sur chaque site. On élabore en parallèle
    a. La définition du produit
    b. Les processus de fabrication
    c. Les processus de maintenance
    d. Les processus d'utilisation
    e. Les processus de fin de vie.

    On est dans le monde du PLM.

Dans ce mécanisme, on va chercher, pour chaque process à exécuter, l'entreprise avec le meilleur savoir-faire dans le monde. L'organigramme fonctionnel n'est plus une donnée initiale, mais est créé dynamiquement pour le programme. On a basculé de l'entreprise intégrée (capable de gérer l'ensemble du processus de conception) à l'entreprise intégrée (dans laquelle le processus est distribué)

Autant, lorsque le travail était fait au sein de la même entreprise, on pouvait avoir une organisation par disciplines (bureau d'études, bureau de calcul, bureau des

méthodes), autant lorsqu'il a fallu distribuer le travail, il était important que les lots de travaux soit un ensemble cohérent en termes de produit, de process et de ressources. Il a fallu trouver une façon de découper le produit qui minimise les interfaces et les couplages entre les composants. Il a enfin fallu prendre en compte les évolutions futures de ces composants, pour être sûr que le produit avait capacité à évoluer dans les prochaines années. Au total, il a fallu mette en place une démarche systémique pour basculer dans ce qu'il est convenu d'appeler l'architecture système.

Alors, les entreprises structurées ont-elles perdu leur avantage concurrentiel. Que nenni. Leur savoir-faire pour gérer l'ensemble du processus de conception reste un véritable atout stratégique. Simplement elles sont passées du statut d'entreprise manufacturière à celui d'entreprise animatrice d'un réseau de coopérants : ce qu'on appelle le métier d'ensemblier.

J'expliquerai cela plus en détail dans les documents

1. SGDT1 – Du fichier à la SGDT
2. SGDT2 – La SGDT (Système de Gestion de Données Techniques)
3. SGDT3 – De la SGDT au PLM
4. SGDT4 – Le PLM, et après ?