La transformation des SI SOA, API, micro-services : comment s'y prendre ?

Apports de la méthode Praxeme pour la conception des architectures de services

Référence PxD-SLB-62 Version 1.0

www.praxademia.com
info@praxademia.com

Intervenant

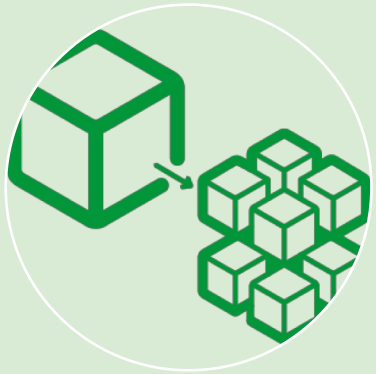
- Dominique VAUQUIER
  - Praxademia S.A.S.
    - Au capital de 40.000 €
    - Créée en septembre 2013
  - Conseil et formation dédiés au développement et à la mise en œuvre de la méthode publique

- Praxeme, méthodologie de transformation d'entreprise
  - Initiative pour une méthode publique
    - Association *Praxeme Institute*
  - Contribution récente
    - Procédé « Assurer la traçabilité de données », sur le *data lineage*
  - Travaux en cours
    - Métrologie de l'entreprise
    - Transformation des SI

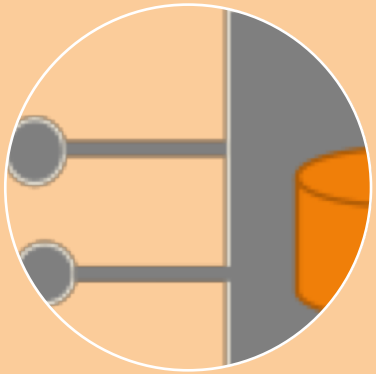
CONIX

Pierre BONNET

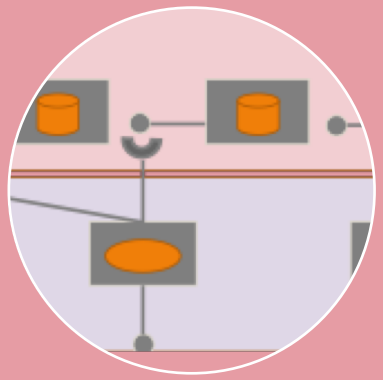
Contenu de la présentation



Que signifie « µService » ?



Quelle est la bonne unité ?

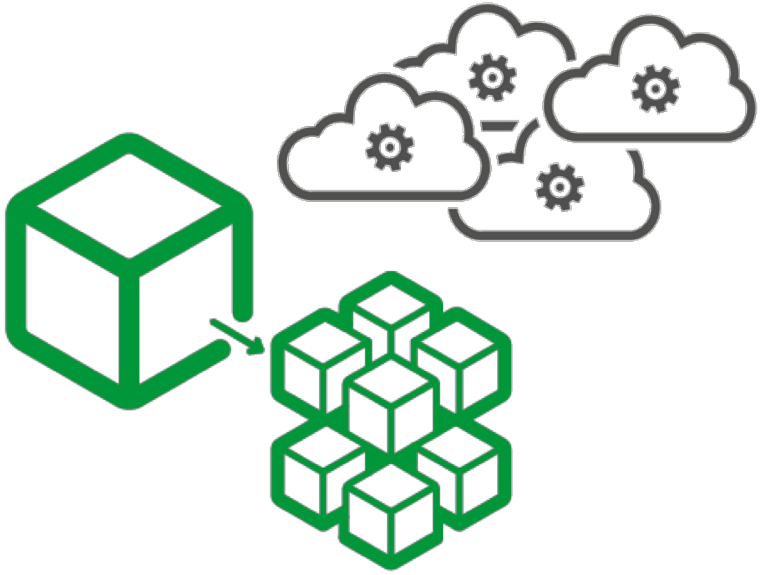


Comment bien structurer le système ?

Synthèse de la méthode

Que signifie « micro-service » ?

- Contenu de la partie
  - La tendance actuelle
  - Différence avec SOA ?
  - ➔ notre interprétation



Point de départ : les tendances

- La généralisation des API
  - *API Management*
  - Les « programmes API » pour améliorer les systèmes existants
- La virtualisation
  - *Cloud computing*
- La « conteneurisation »
  - Faciliter le déploiement
  - En ligne la démarche Devops
    - Déploiement continu
- Les micro-services

Les micro-services : approche de la définition

- Les micro-services… ne sont pas des services !
  - Pas au sens d'un service rendu par le système
    - La métaphore originelle
- Ce sont des composants logiciels
  - À l'instar des Web Services
- Plus précisément, le mouvement « micro-services » pose la question de la structuration du système informatique
  - Il s'agit donc d'un style d'architecture
    - Plus logique que technique
    - Appuyé sur l'offre technologique : API + conteneurs

En quoi un micro-service est-il « micro » ?

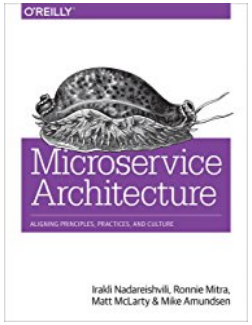
- Par opposition aux silos applicatifs
  - Critique classique
  - Pourquoi les applications monolithiques sont-elles un problème ?
    - Difficulté de déploiement, donc ralentissement dans les évolutions
    - Difficulté à mutualiser, donc redondance garantie
- Par opposition aux « services » d'avant
  - Du temps de SOA (?)

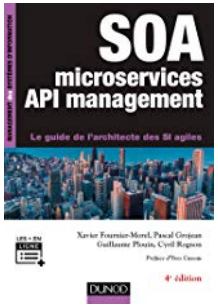
Le style « micro-services » est-il différent du style SOA ?

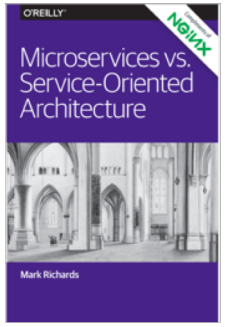
Microservices are the next big thing in modern software architecture and development.

The Microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery.

- (Fowler & Lewis, 2014)

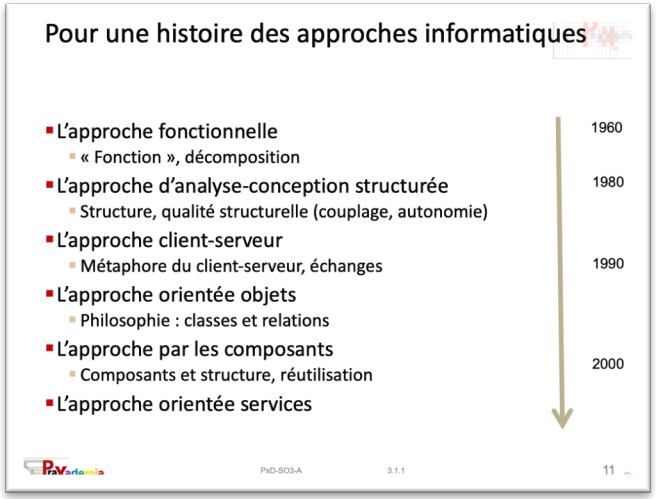






Notre interprétation

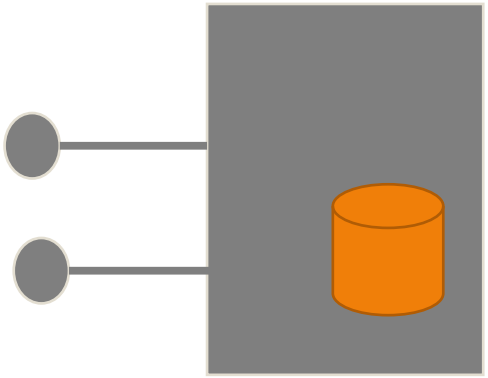
- Les principes revendiqués par le style « micro-services » ne diffèrent en rien des principes au fondement de SOA
  - Du moins, en ce qui concerne l'architecture logique
  - Ce socle de principes résulte d'une longue tradition
- Les différences résident plutôt dans les choix techniques de mise en œuvre
  - Recherche de légèreté
    - *Service mesh* à la place de l'ESB, en gros
  - Conteneurs pour le déploiement
  - API sur des protocoles de base
    - Plutôt que Web Services
  - Solutions *open source*
  - Re



Pour nous, µServices et SOA sont le même style

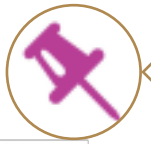
Quelle est la bonne unité ?

- Contenu de la partie
  - Un point critique : l'architecture des données
  - Ce qui change
  - Une terminologie pour architecturer le système
  - L'unité constitutive du système



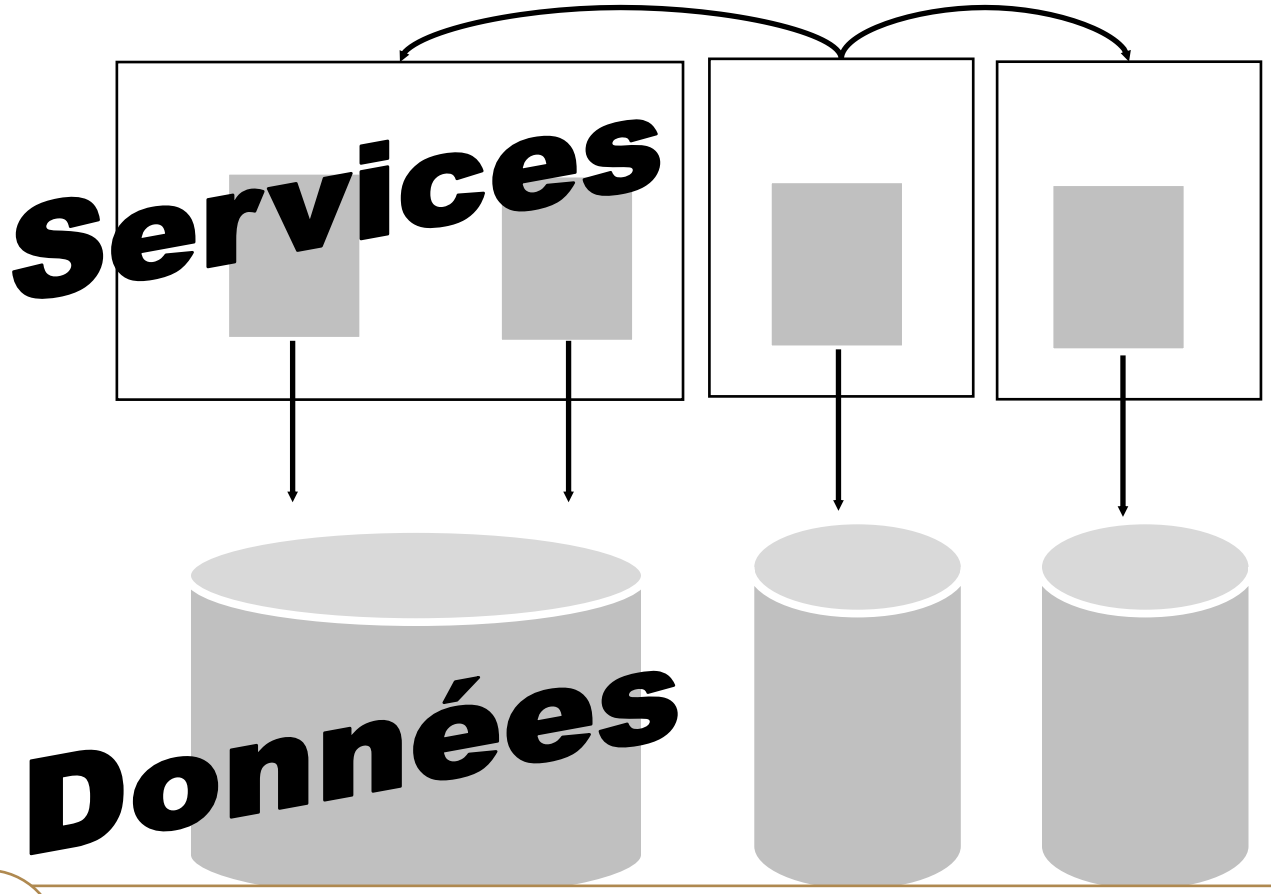
Un point très important : l'architecture des données

- La question est celle du découpage des supports de persistance
  - Cette question, nous l'avons toujours posée dans les projets SOA, depuis le début des années 2000
- Le raisonnement à tenir est simple
  - Diapositive suivante
  - Question posée aux DSI ayant lancé des programmes de transformation de SI en SOA
- Ce raisonnement se heurtait à l'habitude des bases de données très larges
  - Associées à des SGBD puissants



La génération µS fait sauter un verrou !

À quel niveau de constituants logiques faire correspondre les bases de données ?



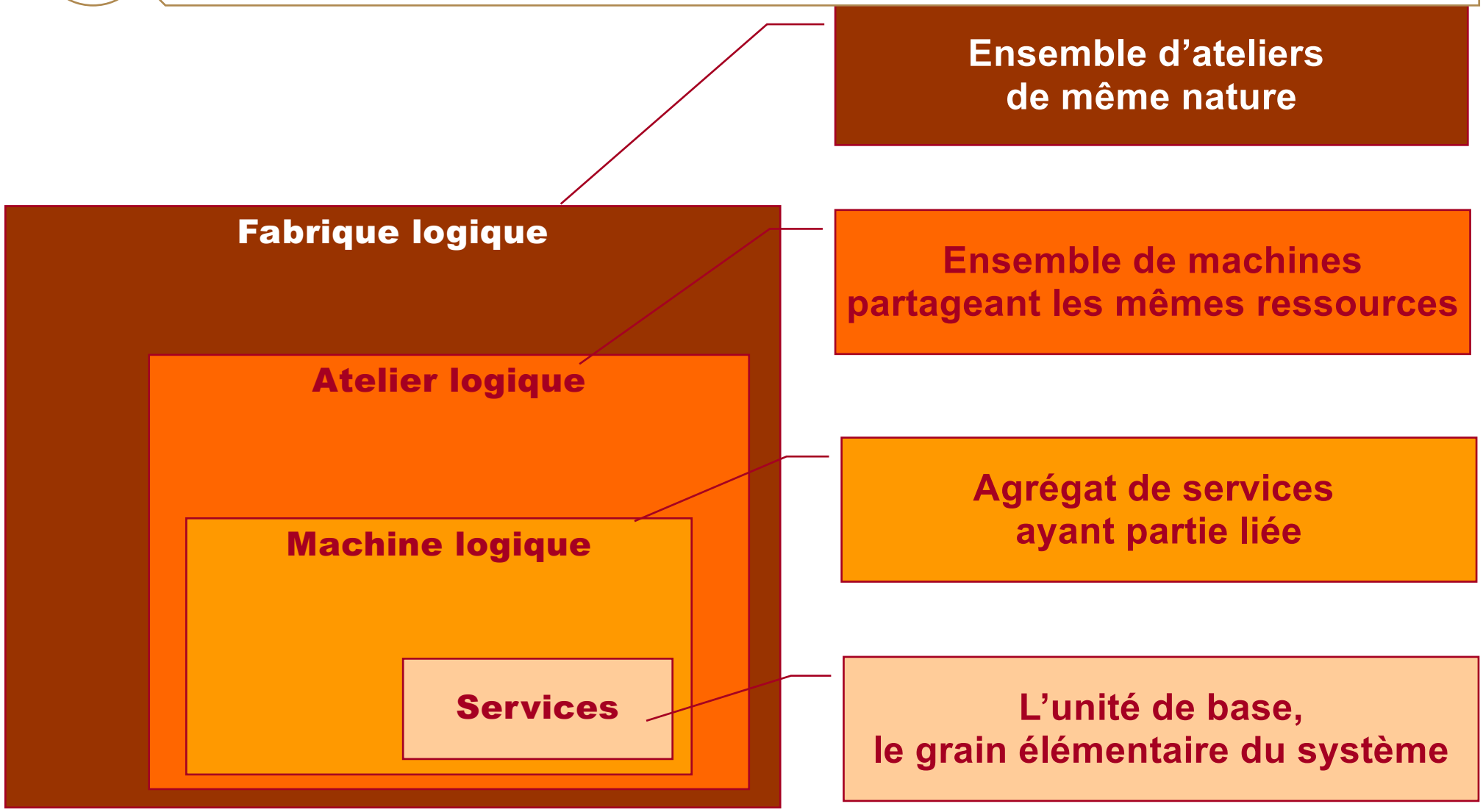
… blocs logiques

Le plan des services surplombe et masque le plan des données

La tendance actuelle milite pour des « petites » bases, encapsulées dans les µServices

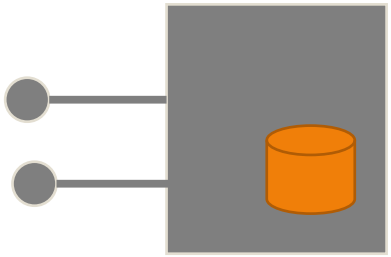
La métaphore pour organiser des services

Terminologie fondée sur l'analogie avec une usine



À la recherche de l'unité constitutive des systèmes techniques

- Deux versants d'un même combat
  - Encapsuler
    - Pour protéger les informations
    - Découpler les composants = augmenter leur autonomie
    - Donc, faciliter les évolutions
  - Exposer
    - En contrepartie de ce qui est caché, fournir les services pour accéder et transformer l'information, sous contrôle
- Représentation formelle
  - UML offre deux notions qui conviennent parfaitement :
    - Le composant
    - L'interface



Représentation symbolique

L’unité qui protège et qui expose : l’atelier logique

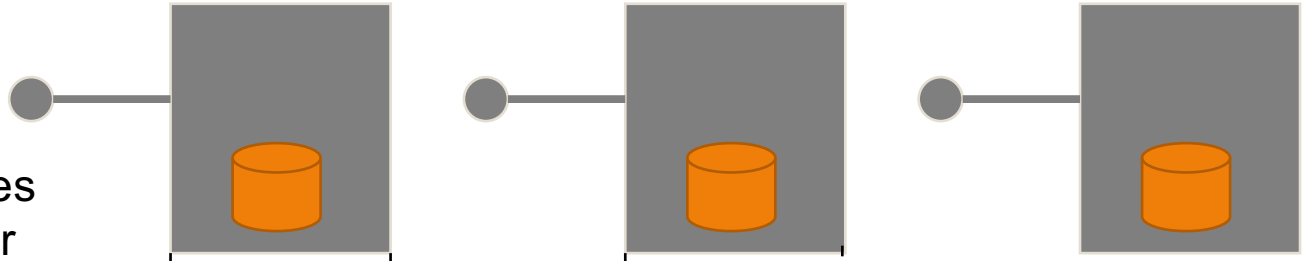
- L’atelier est une boîte noire
  - Principe d’encapsulation
  - Les ressources sont encapsulées dans les ateliers
    - Particulièrement, la base de données ou tout autre solution de persistance
- L’atelier porte les interfaces
  - Les interfaces exposent (et contractualisent) les services
    - En général, 2 à 3 interfaces par atelier
    - On les identifie à partir du point de vue d’ensemble sur le système (architecte)
  - Chaque interface offre une sélection de ce que l’atelier peut fournir
- Dans la suite de la démarche :

L'atelier est l'anticipation du micro-service (logiciel)

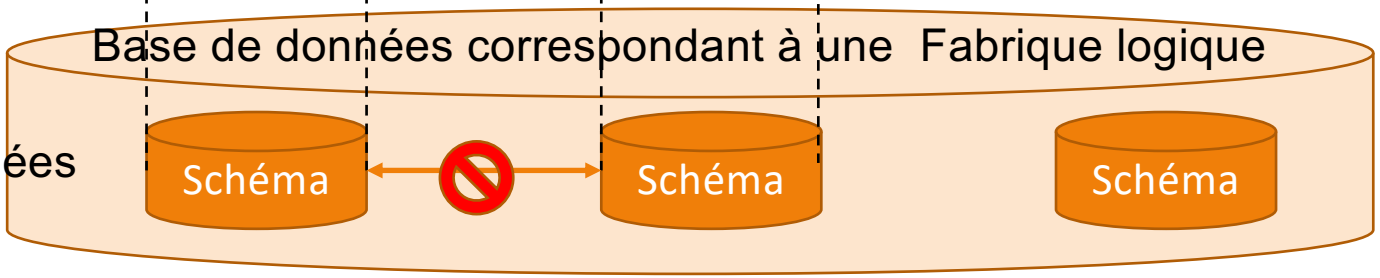
L'atelier logique // unité de déploiement (physique)

Retour sur la question de l'architecture des données

Solution A)
Base de données interne à l'atelier

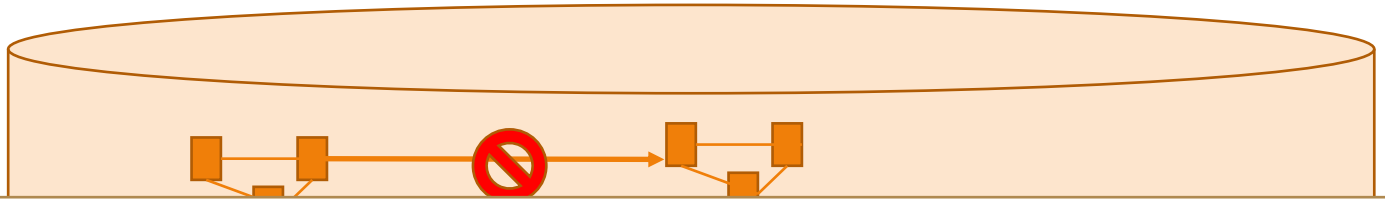


Solution B)
Schéma de données correspondant à l'atelier



Interdiction des jointures entre tables confiées à des ateliers différents

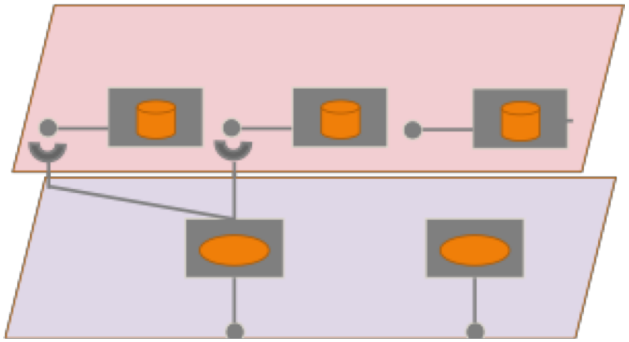
Solution C)
Pas de schéma, mais discipline



La solution A, « pure SOA », devrait devenir la norme avec le µS

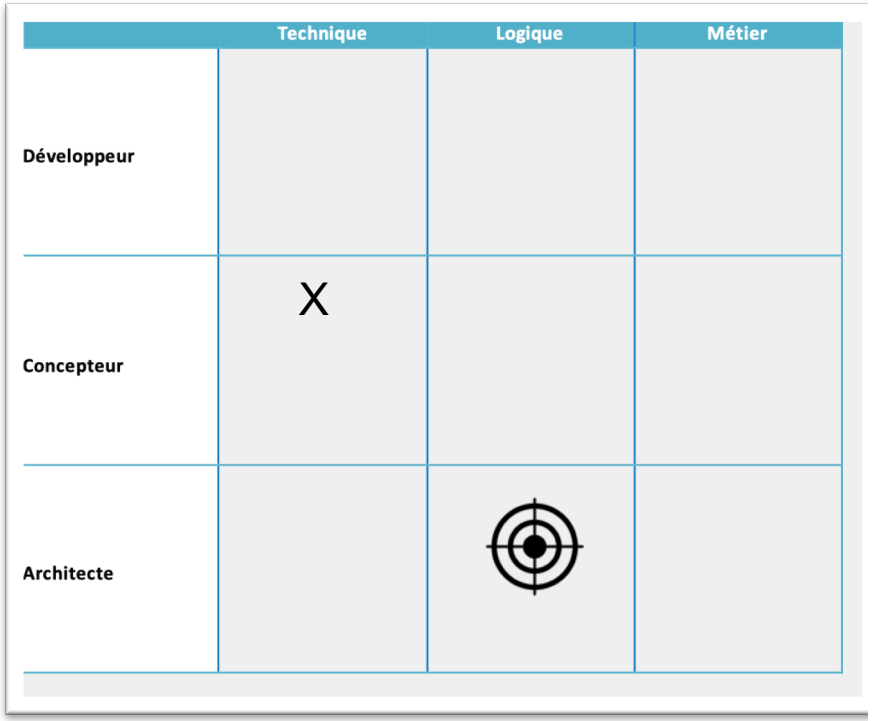
Comment bien structurer le système technique ?

- Contenu de la partie



Préalable : À propos de compétences

- Une fois assurées les conditions de faisabilité technique, il reste à élaborer la structure optimale du système
  - Le partage de responsabilité entre :
    - Architecture technique
    - Architecture logique
  - La négociation logique-technique
    - Un moment dans la démarche pour vérifier les conditions de convertibilité du modèle logique en logiciel
    - Conséquences sur l'expression logique
    - Impact sur le dossier d'architecture technique



Outil d'analyse des compétences

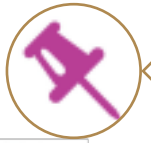
Le raisonnement : point de départ

- Le meilleur système qui soit est celui qui s'aligne parfaitement sur les besoins de l'entreprise et sa stratégie
  - Y compris ses besoins d'agilité, de traçabilité, de qualité, d'interopérabilité, etc.
  - Pour le moindre coût
- D'où la question : comment bien représenter le métier ?
  - Cf. la « séquence fondamentale »
    - Présentations commentées SLB-56 et SLB-57a sur http://wiki.praxeme.org
  - Le métier doit être modélisé en séparant impérativement deux aspects :
    - Aspect pragmatique (ou organisationnel) : les activités métier
    - Aspect sémantique (ou conceptuel) : les objets métier, la connaissance des fondamentaux du métier

Deux points de départ irréductibles pour la conception informatique

Le raisonnement : conséquence sur le système

- Cette séparation entre les deux aspects « métier » devrait être maintenue dans la conception du système technique
  - Les deux aspects évoluent pour des raisons et à des rythmes différents
  - Il en ira de même pour les composantes du système
    - Un noyau stable, composé de constituants construits autour de notions universelles et partageables
    - Une périphérie isolant les évolutions organisationnelles
- La notion de strate traduit cette idée dans l'architecture logique



À l'aube du génie logiciel : « *separation of concerns* »

Les strates et les utilitaires

- Le principe de stratification
  - 3 strates avec contraintes pour la communication
  - Dérogation pour la fabrique logique FL_Utilitaires
    - Où loger différents mécanismes transverses

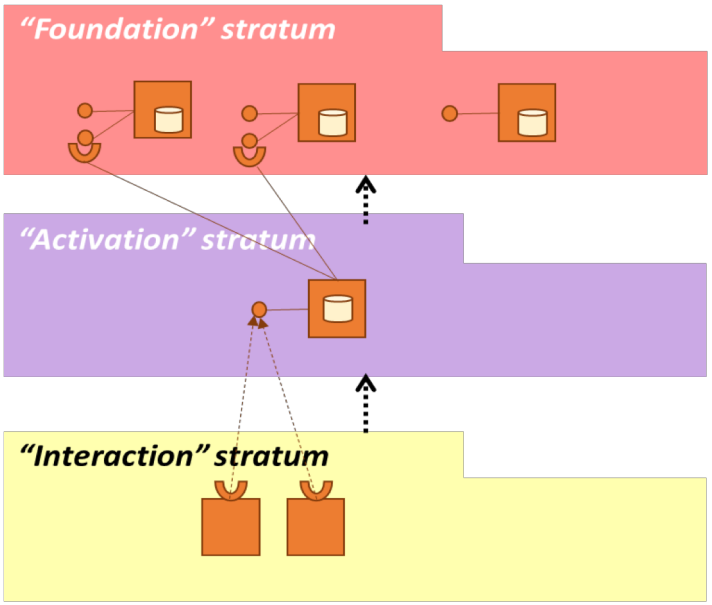


Schéma de principe

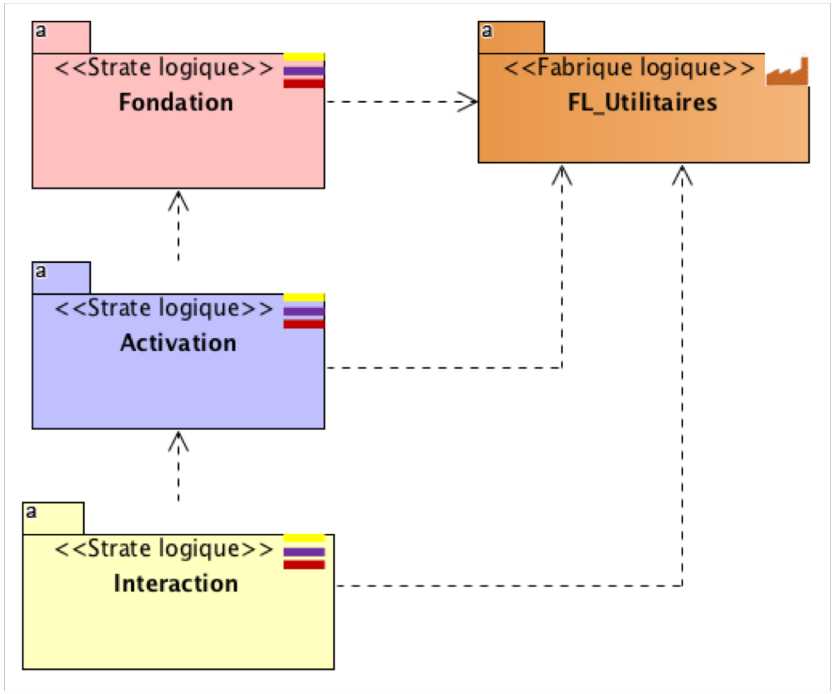


Diagramme UML (Visual Paradigm)

Rappel du cadre de représentation

Aspect sémantique

Les fondamentaux du métier (« objets métier », cycles de vie des objets…)

Aspect pragmatique

Les activités du métier (processus, organisation, situations de travail…)

Aspect logique

L'abstraction du système technique, où se décide la structure optimale (en SOA, la matière du système s'organise par la notion de service et les notions connexes : encapsulation, contrat, interface…)
NB : les services sont trouvés par dérivation des modèles métier.

Aspect logistique

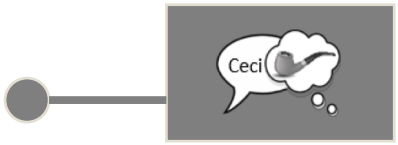
Les types de matériels et les composants logiciels (Les spécifications logiques sont traduites dans les termes des techniques retenues. Éventuellement, plusieurs composants logiciels pour un même constituant logique.)

Le positionnement de l'aspect logique : intermédiaire...

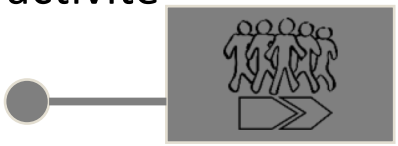
Les dépendances → règles de dérivation

Il y a donc deux catégories d'ateliers logiques

- On les distingue selon :
  a) leur origine dans les aspects « amont »
  b) leur situation dans l'architecture logique
- Le constituant de la strate « Fondation »
  - Déduit de l'aspect logique
    - Dérivé d'un sous-domaine d'objets



- Le constituant de la strate « Activation »
  - Déduit de l'aspect pragmatique
    - Dérivé d'un domaine d'activité



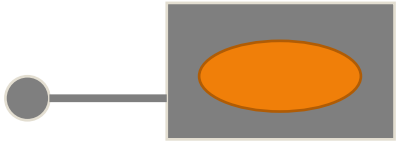
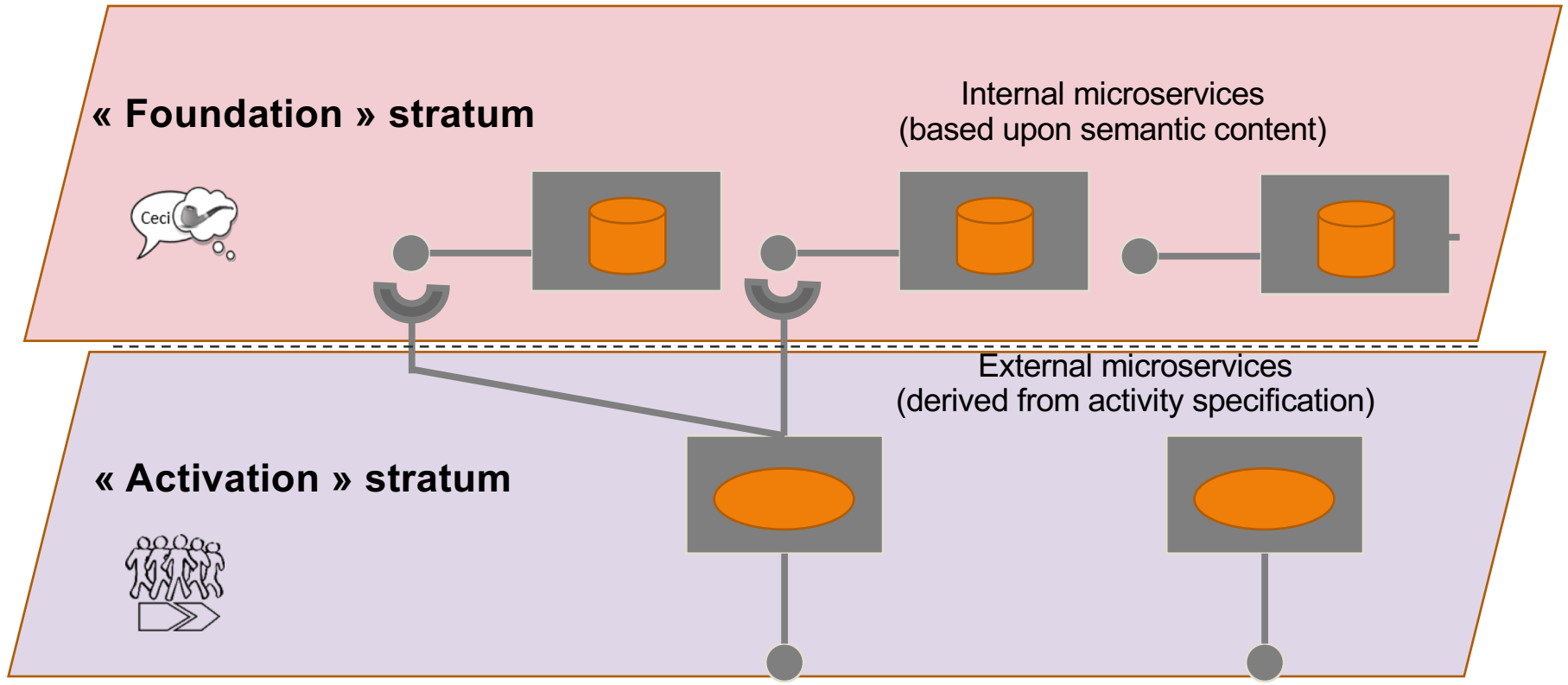


Schéma de principe de l'architecture logique : premier aperçu



Règle 1 (absolue) : polarisation de la communication

Règle 2 : réduction du couplage (au moins, latéral)

Illustration : vue synthétique du métier

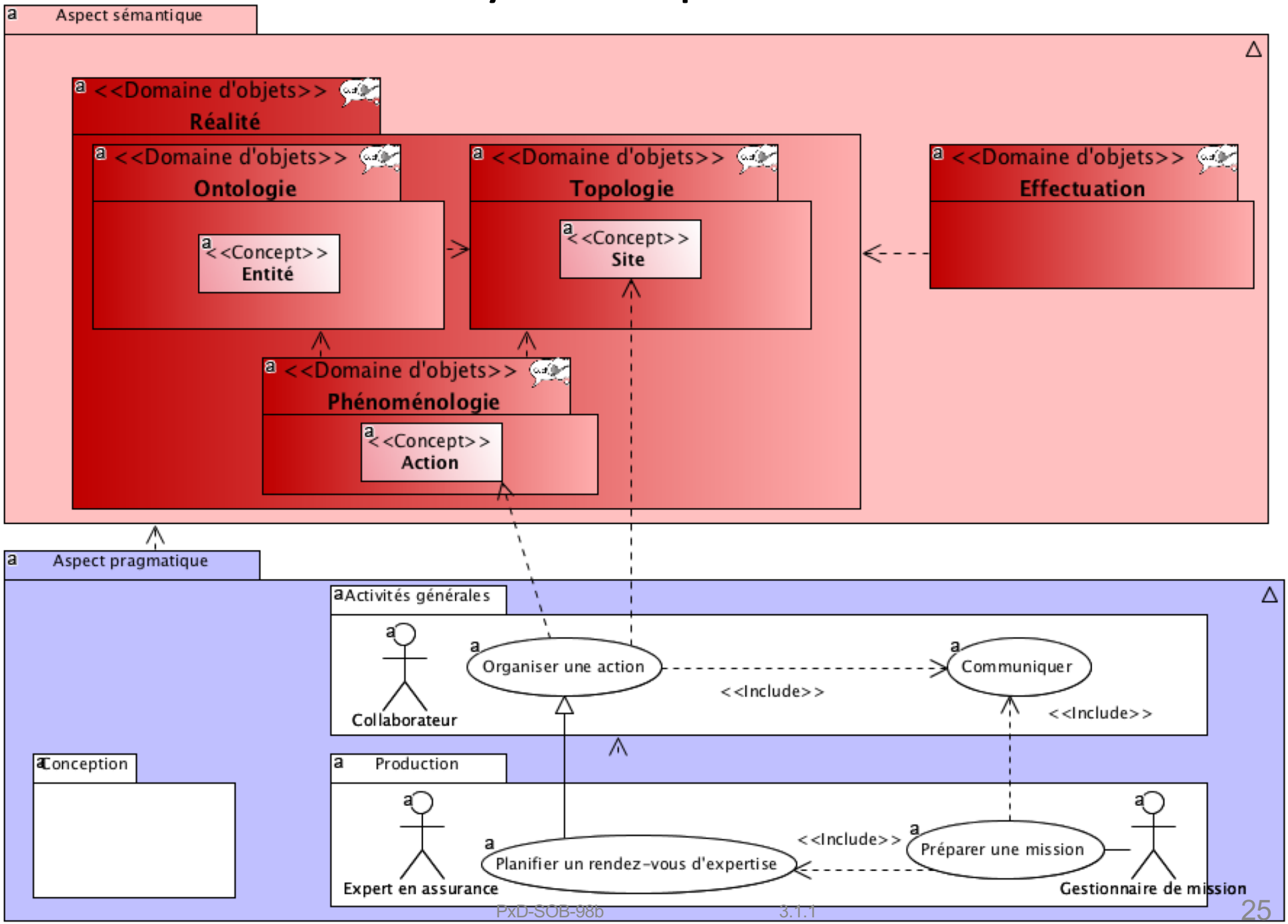


Illustration: Le graphe d'architecture logique de niveau 3 ➔ les ateliers logiques de « Fondation »

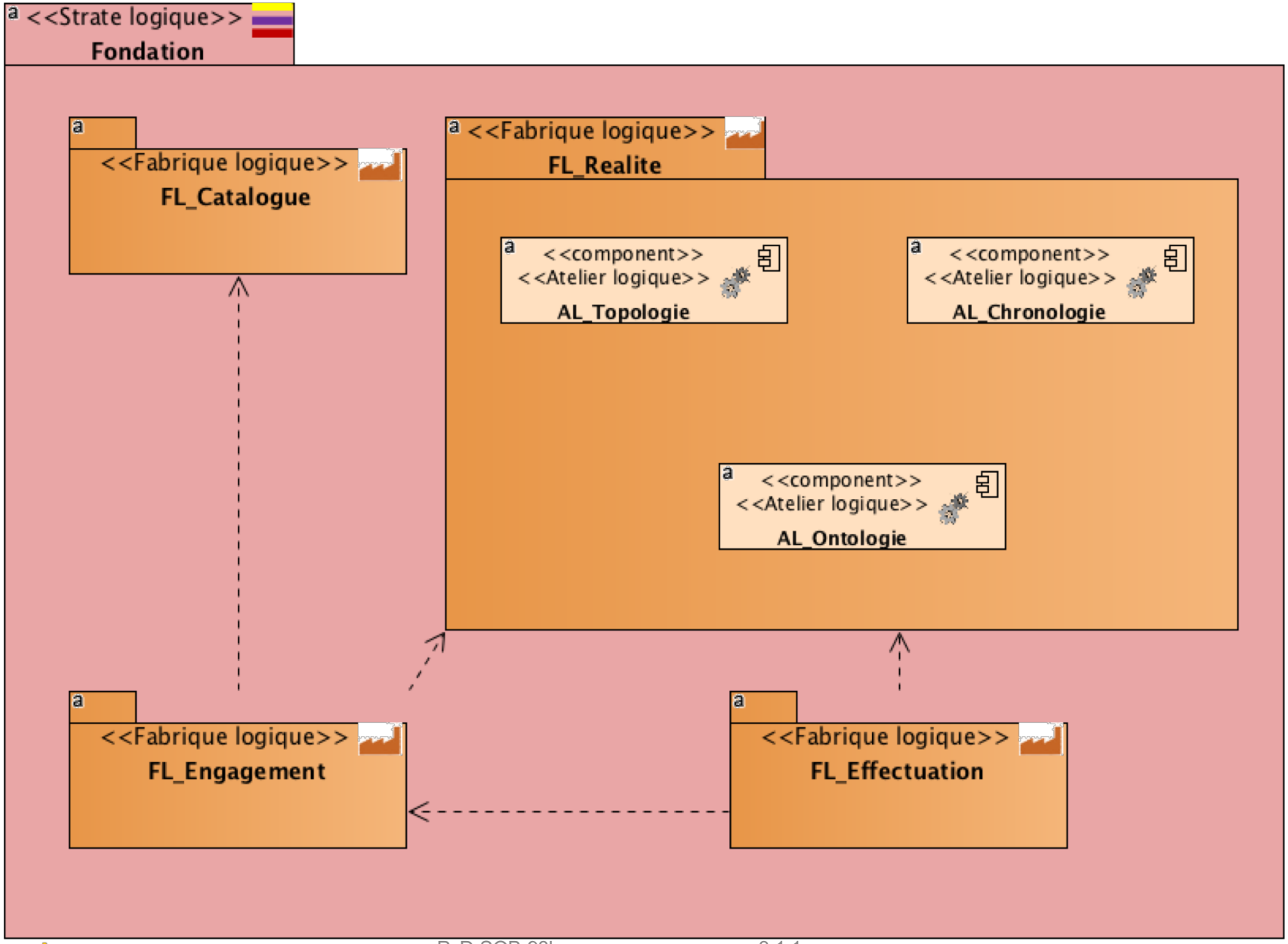


Illustration : graphe d'architecture logique dans le périmètre d'un projet

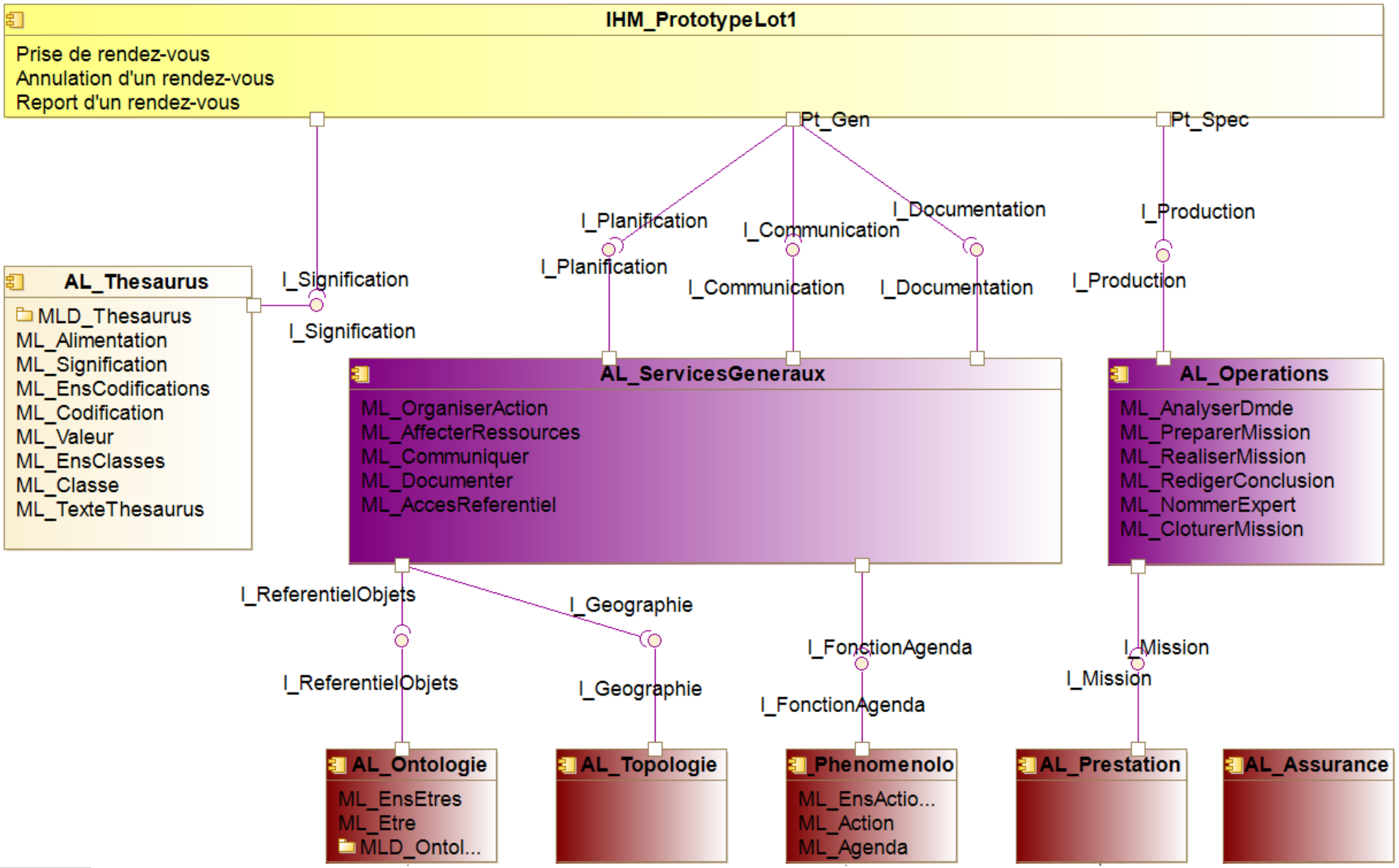
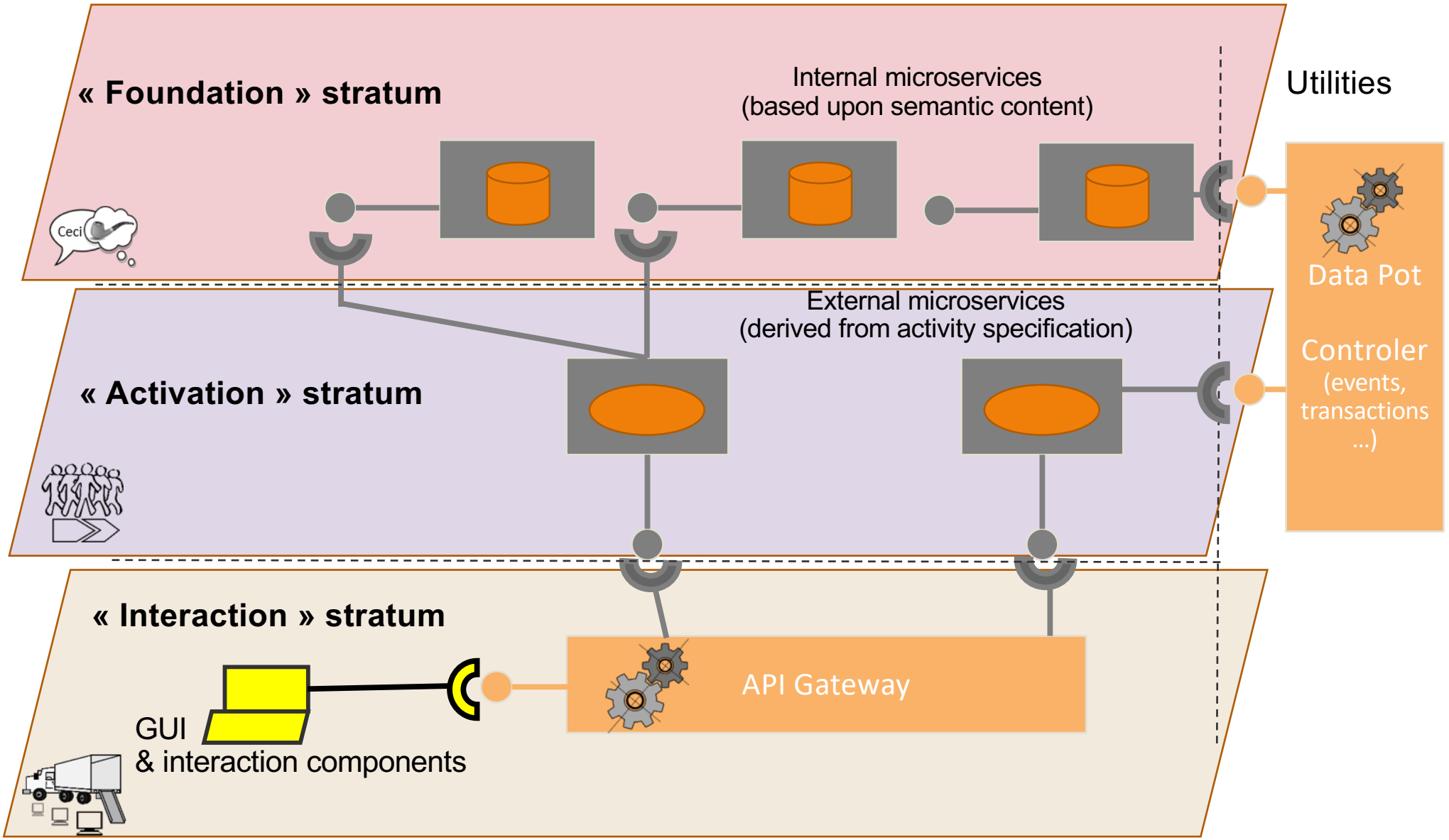
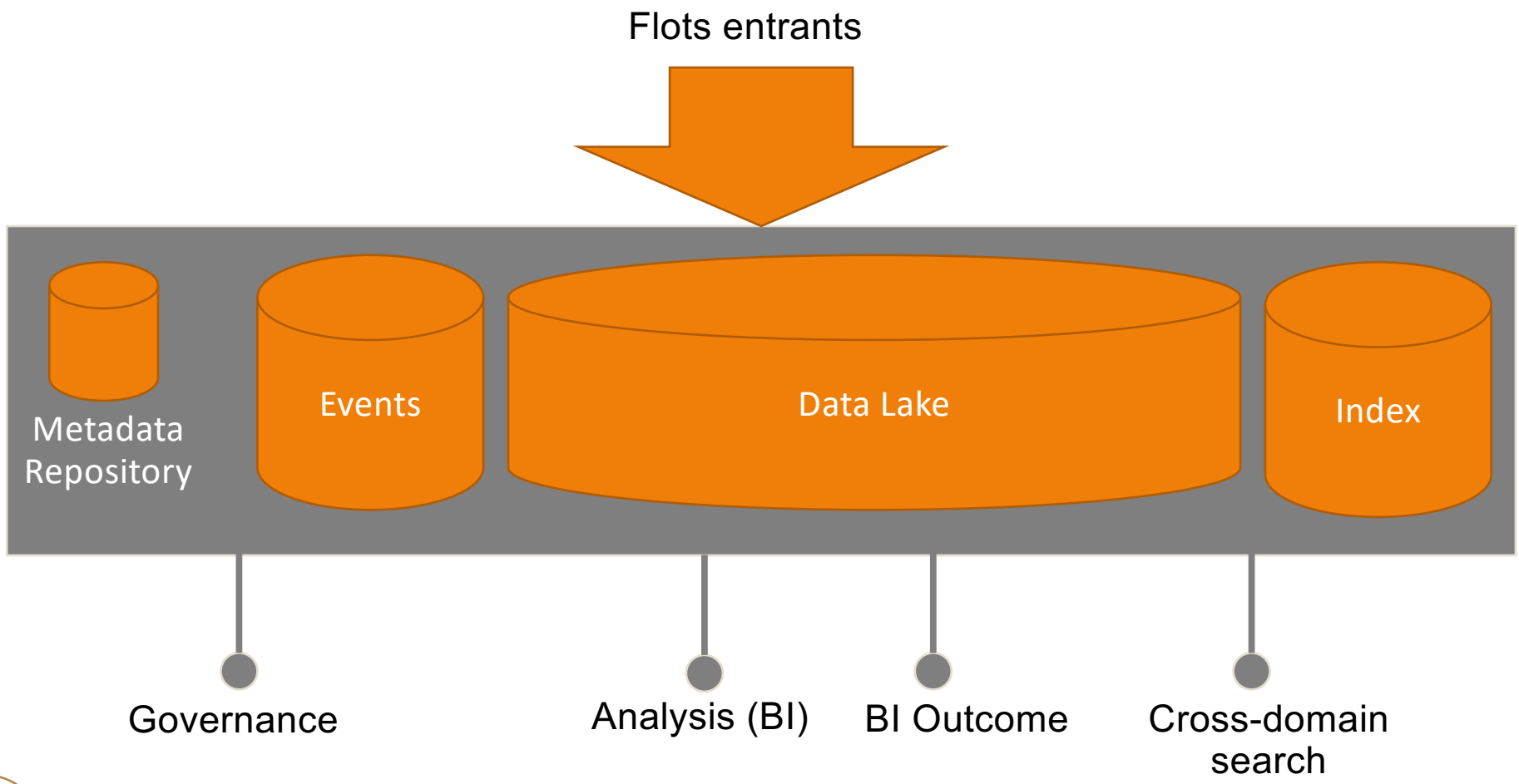


Schéma de principe : complété



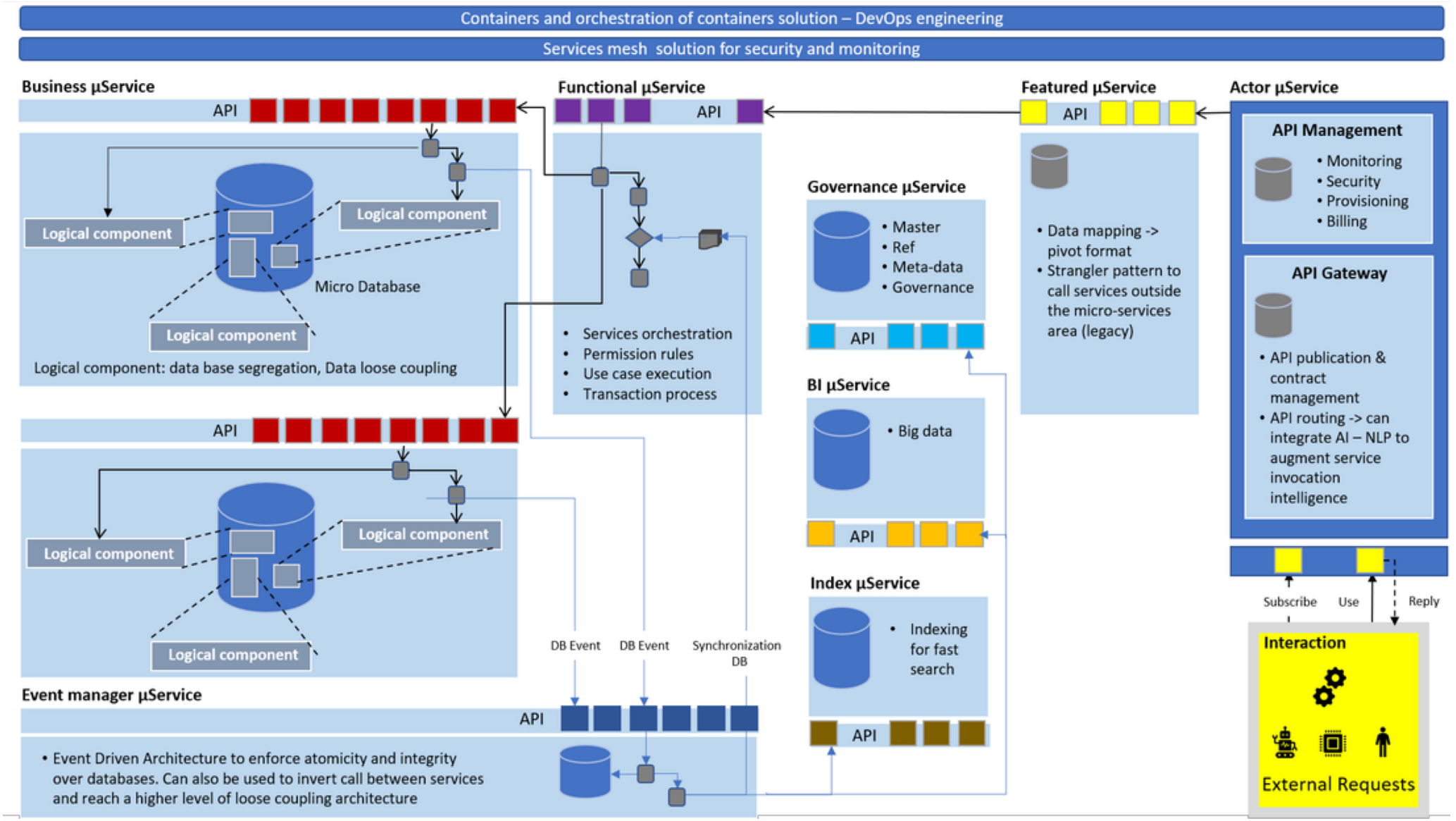
Le *Data Pot* : la doublure du système



Plutôt une fabrique logique

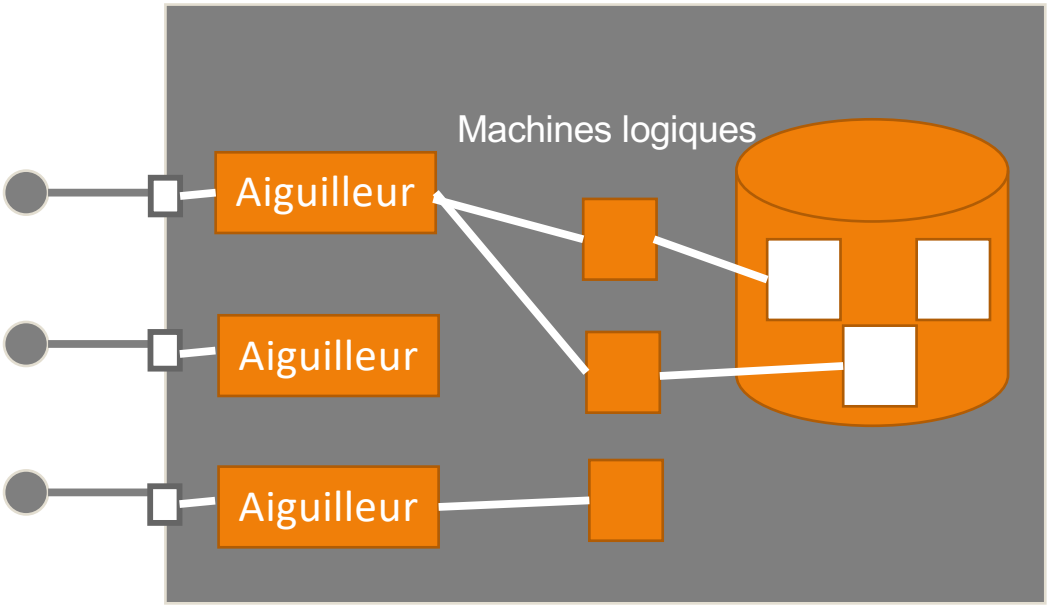
Vu de l'extérieur, ce qui compte, ce sont les interfaces

Version de Pierre BONNET (Smart-Up)



L'architecture logique se prolonge par la conception logique

- Contenu d'un atelier
  - La spécification logique des services peut / doit aller jusqu'aux algorithmes
    - Recours à un pseudo-langage pour une spécification rigoureuse, susceptible d'être passée aux programmeurs



Avec quelques règles

- Deux à trois interfaces fournies par atelier logique
  - Interface définie en termes de mandat par rapport à un usage
  - L'interface expose les services
    - Les services sont représentés par des opérations
    - L'interface est le protocole de communication, comprenant les signatures et les contrats de service
- Un aiguilleur par interface
  - Il fait le relai entre l'interface (extérieur) et les machines où se trouvent les opérations et les algorithmes (services internes)
- Pour chaque table
  - Une machine ensembliste
  - Une machine élémentaire
- Contraintes topologiques
  - Respect des règles de communication, limitation du couplage…

Illustration: une interface

- Trois notions à distinguer
  - Interface protocolaire : services exposés et contrats
  - Interface fournie 
  - Interface requise 
- Exemple
  - Les services exposés pour manipuler les objets Actions

I_FonctionAgenda
+ decrireAction(in action: InfoAction, out codeRetour: InfoSignal): InfoAction + rechercherActions(in borneMin: InfoAction, in borneMax: InfoAction, inout pagination: InfoPagination, out codeRetour: InfoSignal): Inf... + affecterRessource(in action: InfoAction, in participation: InfoParticipationAction, out codeRetour: InfoSignal): InfoAction + fixerMoment(in action: InfoAction, in periode: InfoPeriode, out codeRetour: InfoSignal): InfoAction + indiquerProprietaire(in action: InfoAction, in rfProprietaire: IdObjetType, out codeRetour: InfoSignal): InfoAction + fournirAgenda(in rfRessource: IdObjetType, in periode: InfoPeriode, in cdDisponibilite: string, out codeRetour: InfoSignal): InfoAgenda + calculerCreneaux(in action: InfoAction, out codeRetour: InfoSignal): InfoAction + liberer(in action: InfoAction, in options: InfoPeriode [*], in cdProgrammation: string, out codeRetour: InfoSignal): InfoAction

Illustration: la construction des services d'activation

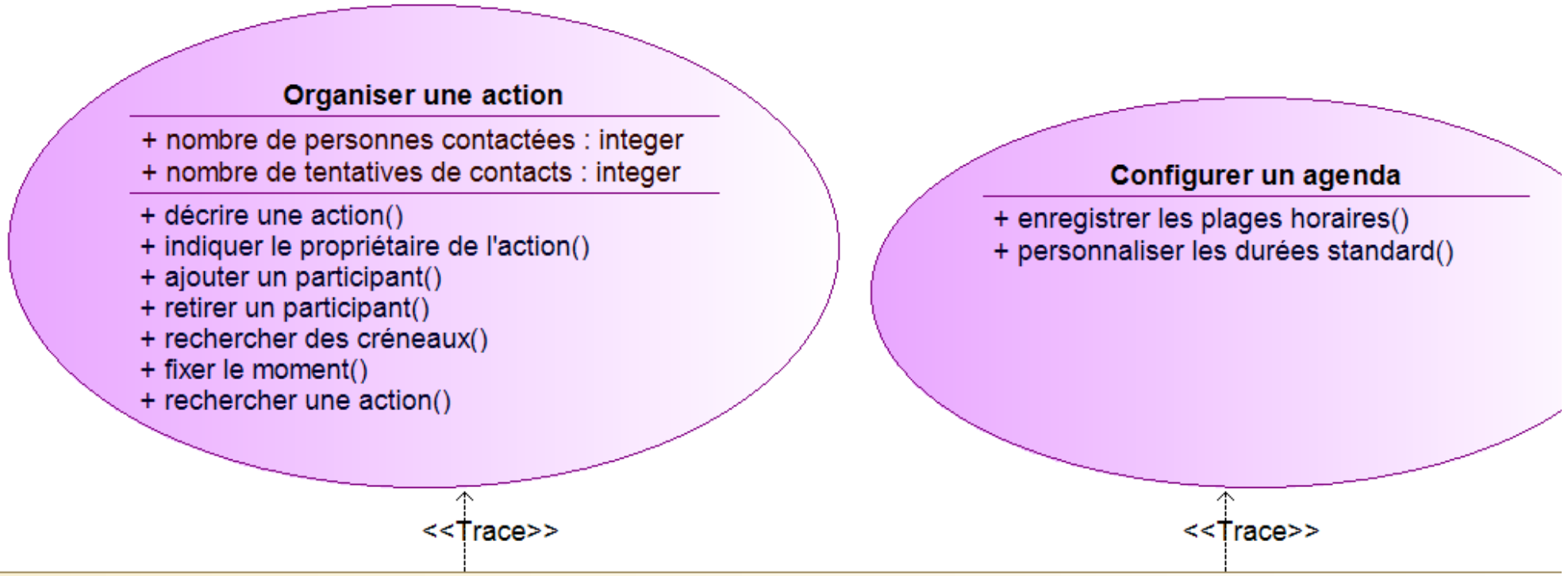
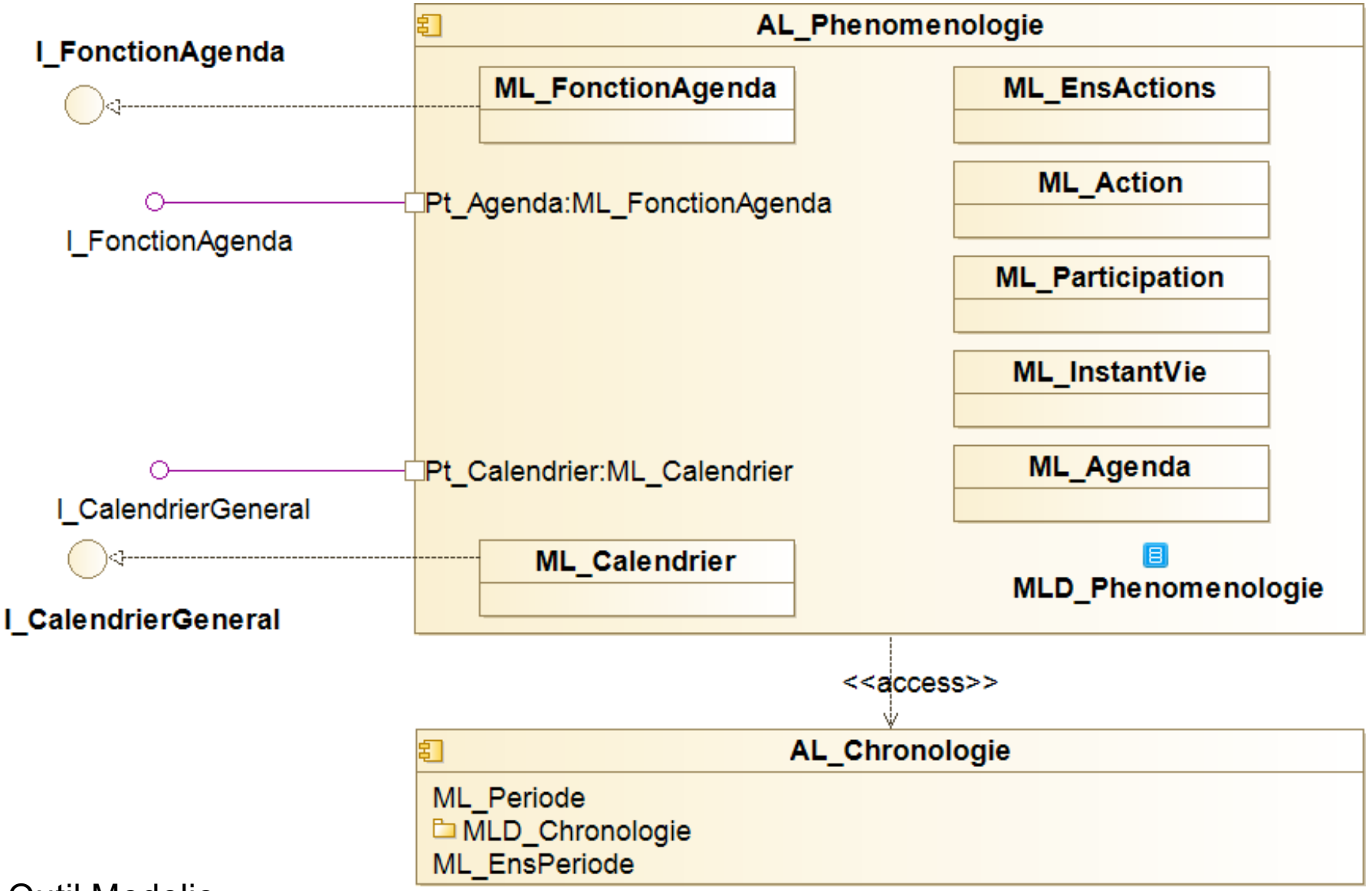


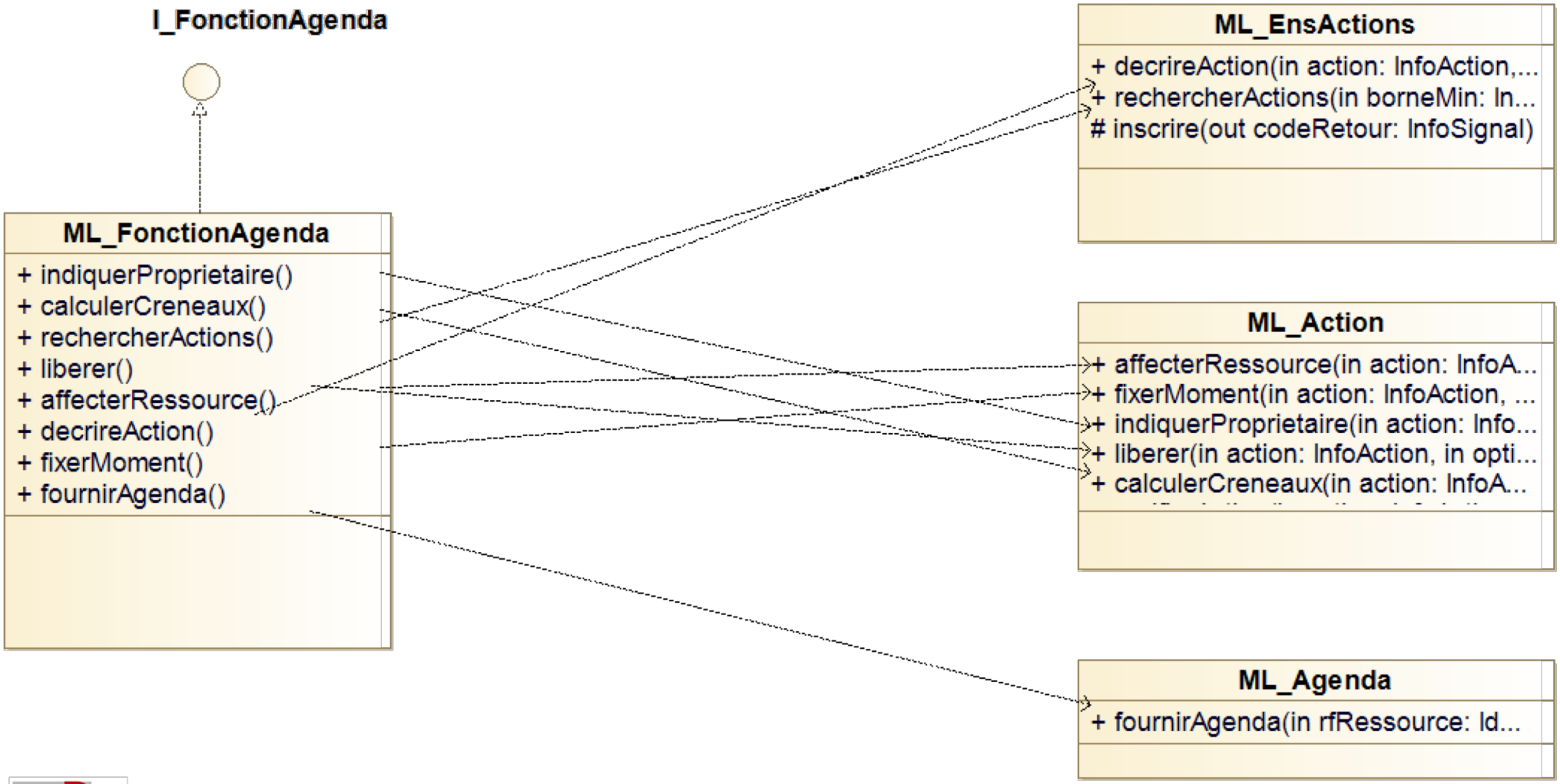
Illustration : le contenu d'un atelier logique



Outil Modelio

Illustration : l'aiguillage, à l'intérieur d'un atelier

- Essentiellement un relai vers les services internes à l'atelier



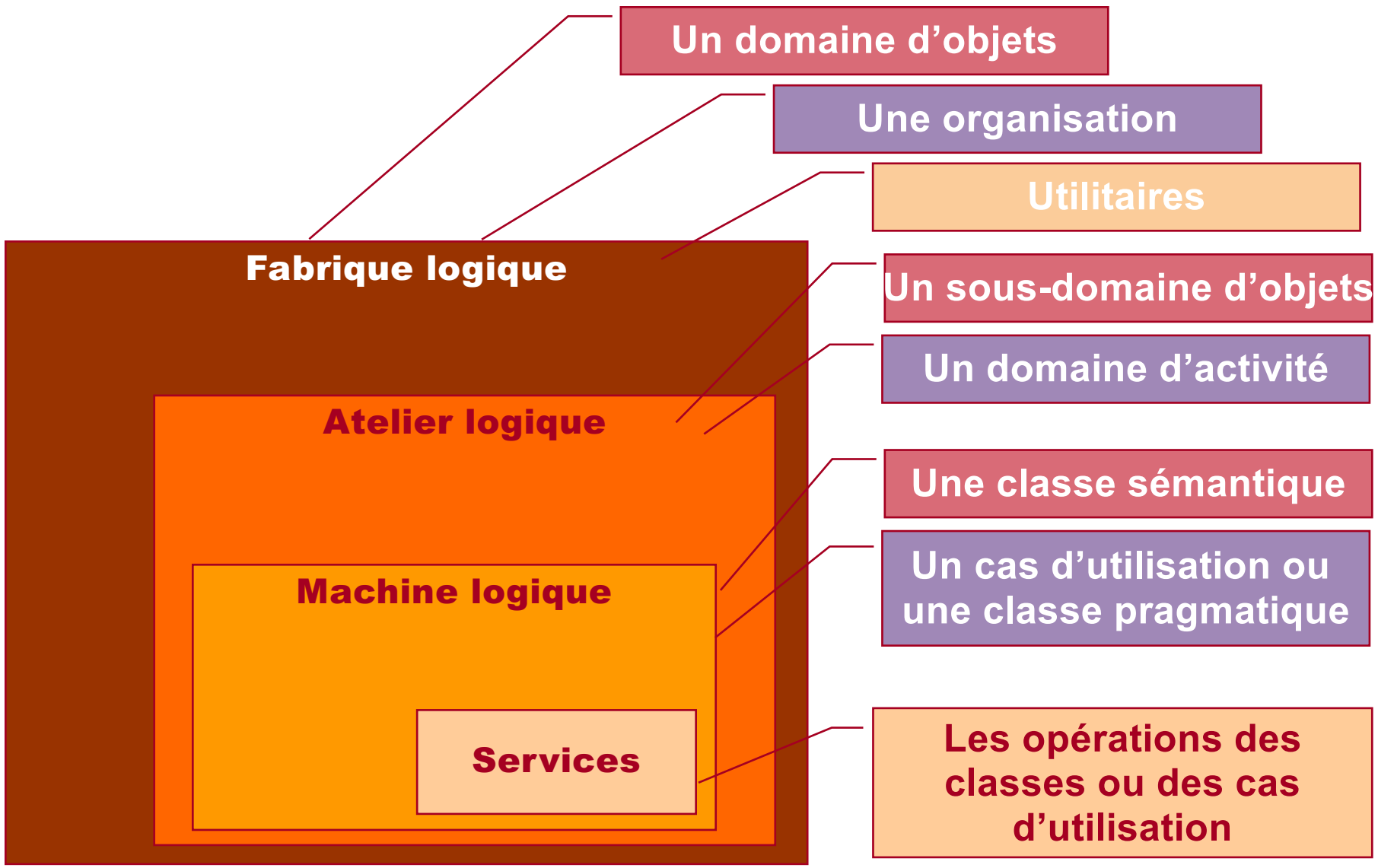
Synthèse de la méthode

- Contenu de la partie
  - Vocabulaire de l'aspect logique
  - Notion de facette
  - Cadre de représentation

Vocabulaire

- Un micro-service n'est pas un service !
  - Pas plus qu'un Web Service
  - Ce sont des composants
    - Quand un consommateur sollicite un service, il ne demande pas le composant !
    - Il demande au composant de lui faire quelque chose, de lui rendre un service
- Le service correspond donc à l'opération exposée et effectuée par le composant
  - Dans Praxeme, la terminologie de l'aspect logique respecte la métaphore du service
    - Techniquement, « Service » est un stéréotype UML sur une opération
    - Service exposé : opération sur une interface (interface protocolaire)
    - Service interne à l'atelier : opération sur une machine logique

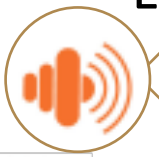
L’origine des constituants logique, par dérivation



Les présupposés à la conception logique

- En entrée : des modèles métier de bonne qualité
  - Respectant la séparation absolue entre sémantique et pragmatique
    - Objets métier et activités métier
    - Qualification des règles métier
  - Complets et suffisamment précis
    - Opérations et automates sur les classes et les cas d'utilisation
    - Notamment, projection des indicateurs de performance
  - Correctement structurés
    - La redondance doit avoir été éliminée de la représentation du métier
    - Dans le cas contraire, elle sera transportée dans la solution informatique
- Des moyens d'expression qui trouveront une traduction
  - Le moment de la négociation logique-technique

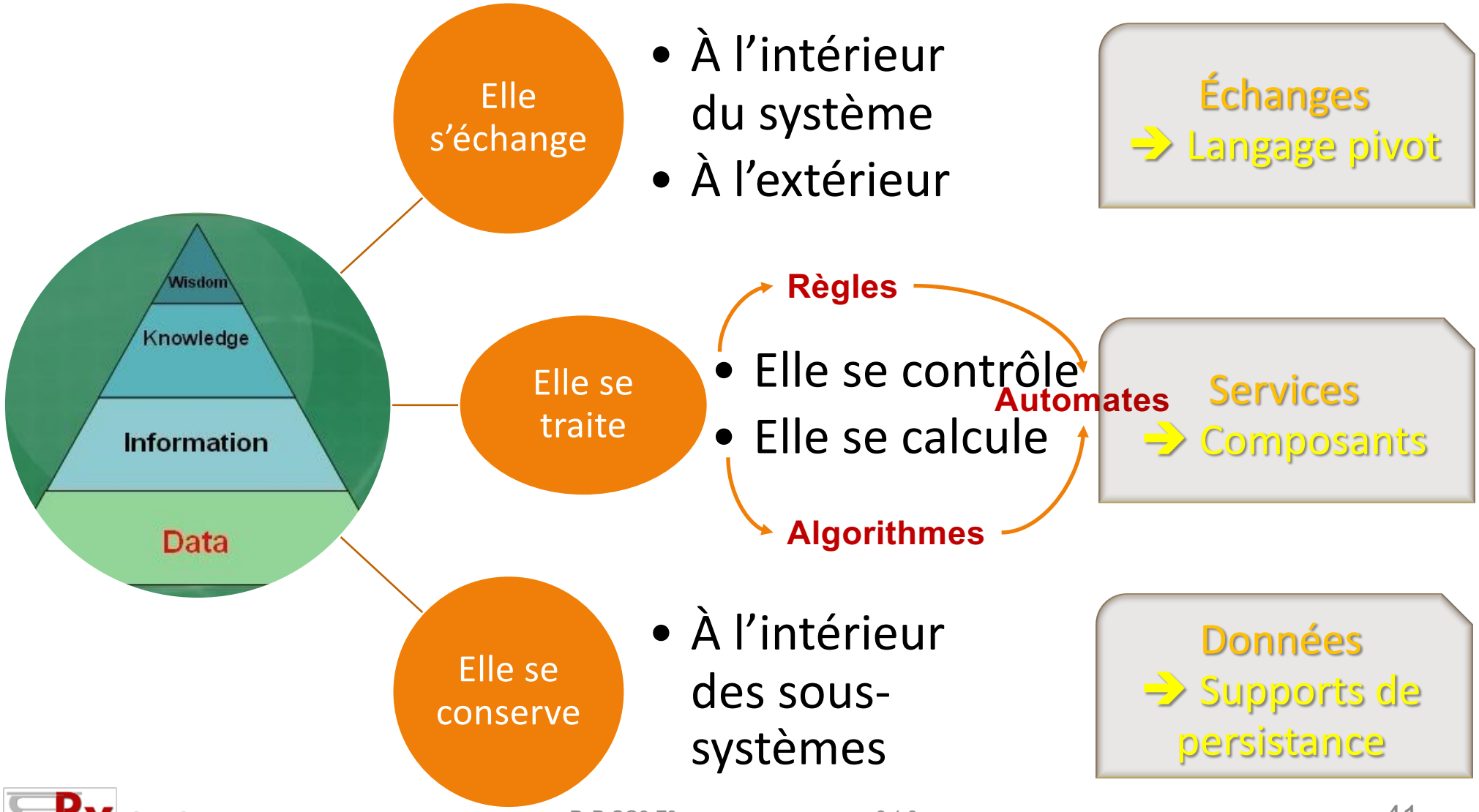
Placer la conception sous l'autorité de l'architecture



Restaurer les disciplines de l'aspect logique

Les trois facettes de l'aspect logique

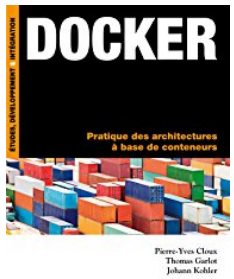
- La matière première – la substance – est l'information



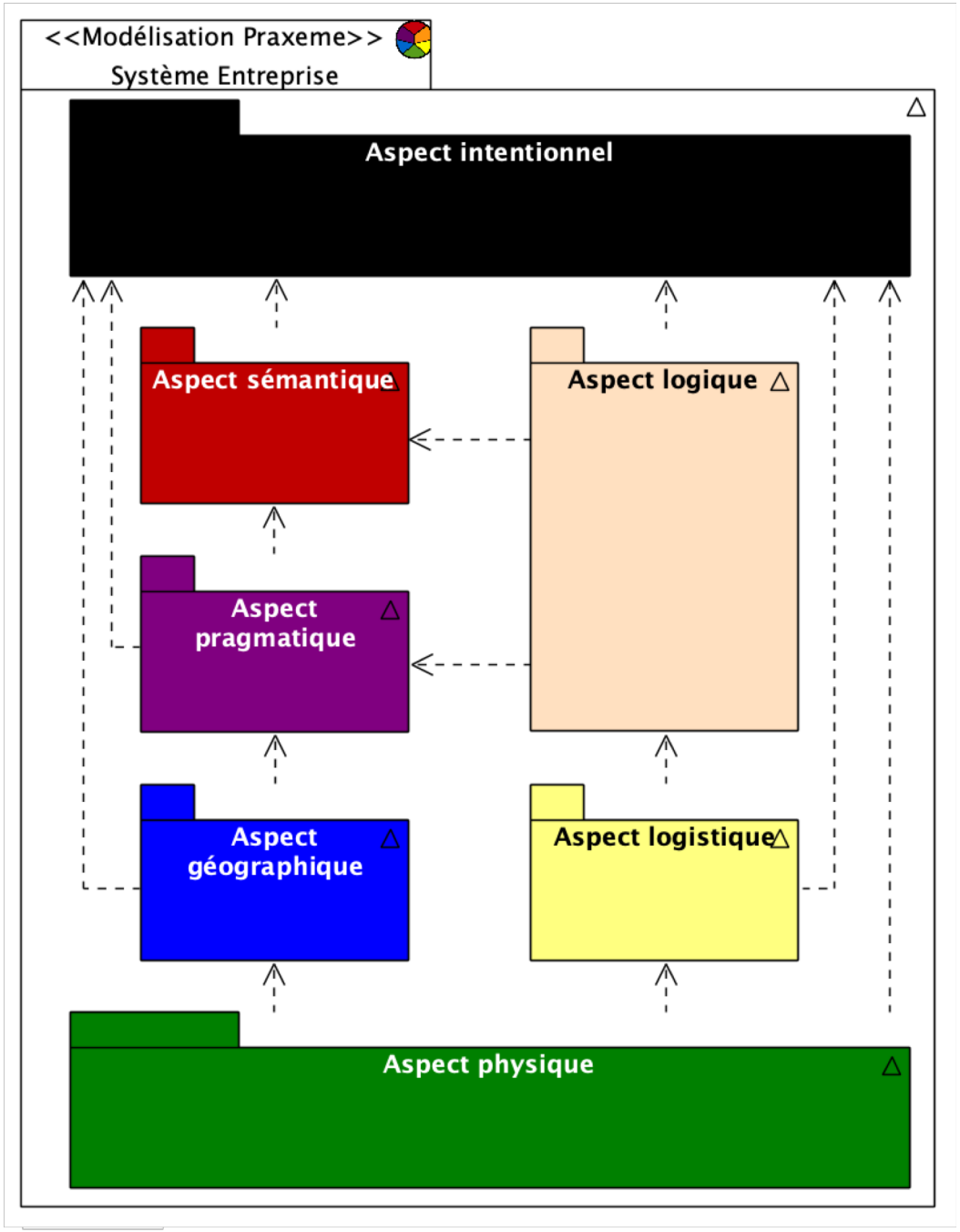
Ce qui se passe après la conception logique

- Aspect logistique (dont l'informatique)
  - Développement du composant logiciel (le micro-service)
    - Traduction de la spécification logique dans une architecture technique cible
    - Éventuellement, plusieurs cibles techniques (surtout pour les composants d'interaction, à la périphérie du système)
- Aspect physique
  - Instanciation du composant et localisation dans la géographie de l'entreprise
    - Y compris *cloud computing*
  - Unité de déploiement au niveau physique : le conteneur
    - « Un conteneur est tout simplement un système de fichiers sur lequel s'exécutent des processus (de préférence un par conteneur) de manière :
      - Contrainte : Grâce à Cgroups qui spécifie les limites en termes de ressources
      - Isolée : Grâce notamment à Namespaces qui fait en sorte que les conteneurs ne se voient pas les uns les autres »

Source

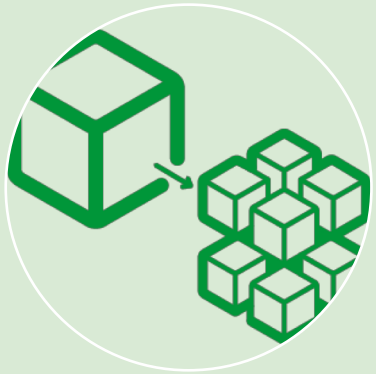


Besoin d'un cadre de représentation complet



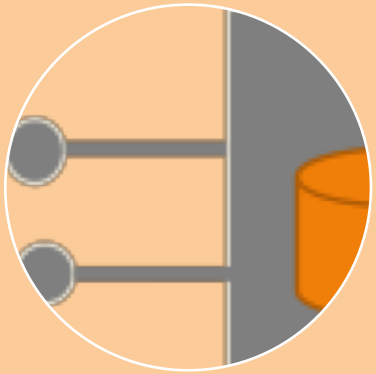
- Pour appuyer cette approche holistique de l'entreprise
  - Structure du référentiel de description de l'entreprise
    - 7 aspects
    - Procédés par aspects
    - Dépendances entre les aspects
    - Règles de passage : projection et dérivation

Conclusion



Que signifie « µService » ?

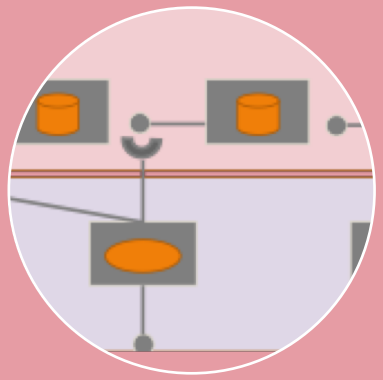
Réactivation de l'approche SOA (style logique)
➔ Une nouvelle opportunité pour bien faire



Quelle est la bonne unité ?

L'atelier logique

(terminologie appuyée sur la métaphore du service)



Comment bien structurer le système ?

Dérivation des modèles du métier
Stratification, architecture et conception logique

Conclusion

- Les questions posées
  - Le style d'architecture SOA a été défini au début des années 1990. Est-il toujours d'actualité ?
  - Avec l'API Management et les micro-services, qu'est-ce qui change ?
  - Derrière les communications et les rumeurs, quels sont les notions et les principes qui doivent guider l'amélioration des systèmes informatiques ?
  - En quoi ces approches contribuent-elles aux investissements sur les solutions « numériques » ?

À retenir — La transformation des SI – Comment…

- La vague µService réactive l'approche SOA
- Investir sur les compétences logiques rend plus efficace la gestion des compétences

Annonces

Voir sur le site http://www.praxeme.org

- Concernant le thème des micro-services
  - « *µServices & SOA* » par Pierre BONNET
    - À paraître sur http://www.smart-up.org
  - « *A method to set up the microservice architectural style* » par DVAU
    - Disponible sur : http://www.praxademia.com (publications en anglais ; SLB-61)
- Concernant SOA
  - Sur wiki.praxeme.org : guide version 1 et articles SLB-15, 50, 58…
- Formation
  - « SOA, conception d'une architecture de services », version 3
- Travaux envisagés cette année
  - Thématique technique
  - Guide et procédés de l'aspect logique en version 2
  - Contribution 2019 de CONIX