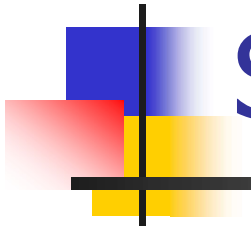
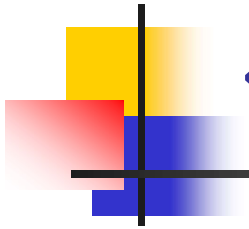


Cours d'architecture logicielle
Styles architecturaux

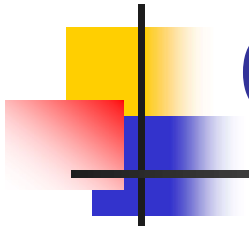
Lydie du Bousquet
Lydie.du-bousquet@imag.fr

Philippe Lalanda



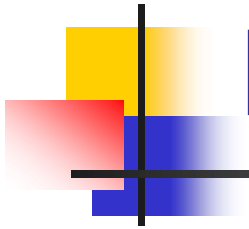
Vous avez dit « Style architectural » ?

- Exercice GCC
  « Pipe and filter »
- Exercice OSI
  « Layered » (en couches)



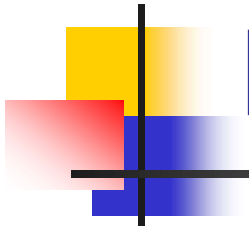
Objectif de ce cours

- Définir la notion de style architectural
- Présenter des styles connus
- Développer sur des exemples



Design Pattern / Patron de **conception**

- Solution à un problème dans un contexte donné
- Il abstrait et documente d'une façon compacte :
  - le problème
  - le contexte
  - une bonne solution
- Basé sur une expérience montrant pourquoi la solution répond bien au problème
- « Pattern Language » : un ensemble de patterns travaillant ensemble
  - Exemples : patterns pour le temps réel, pour la distribution, …

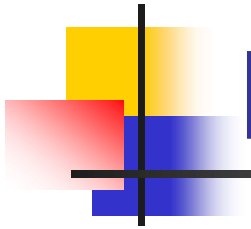


Patron et abstraction

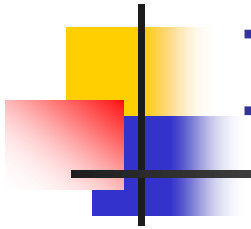
Des patrons à différents niveaux d'abstraction

- « basic building blocks »
  (algorithmes, composants)
  - hash tables, listes chaînées, algorithmes de tris, …
  - encapsulation, héritage et polymorphisme, …
- Design Patterns : conception détaillée
  - patterns pour l'objet
- Styles architecturaux : conception générale
  - Exemple : architecture 3-tiers

Styles architecturaux = patrons d'architecture

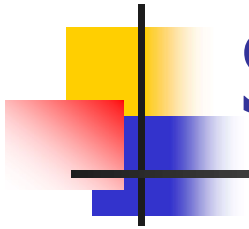


- Un style architectural est un modèle d'architecture définissant
  - des types de composants et de connecteurs
  - une topologie
  - un type de contrôle et de flux de données
  - des contraintes
  - un vocabulaire associé
- Tous ces éléments sont nommés et véhiculent une sémantique



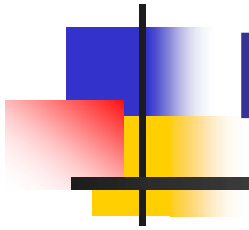
Importance des styles

- Les styles architecturaux sont très importants.
- Ils permettent
  - de lancer la phase de conception
  - de guider les évaluations
  - de définir des outils performants
- Ils fournissent des assurances de qualité
- Limites actuelles
  - encore peu de catalogues
  - encore assez hétérogènes
  - manque de description qualitative des styles (attributs)

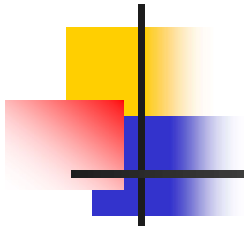


Sondage

- Connaissez-vous le style « client-serveur » ?
  - Oui très bien
  - Oui vaguement
  - Non
- Connaissez-vous le style « MVC » ?
  - Oui très bien
  - Oui vaguement
  - Non



Exercice 2 sur Moodle

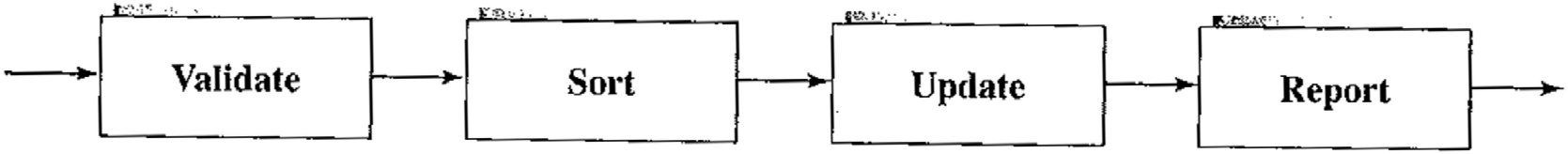


Exercice 3 – par équipe

- Chaque membre de l'équipe choisit un style parmi les suivants
  - 3-tiers, layered, Pipe and filter
- Après avoir étudié un style, expliquez-le aux autres
- Ensemble et pour chaque style, remplissez le tableau suivant
- Répondez au quizz 3 (individuel)

Style :

Principes	
Caractérisation	
Avantages	
limites	



Style : Pipes and filters

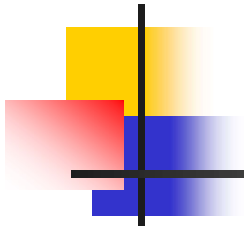
Principes	Transformations des données (data-flow)
Caractérisation	Composants = filters (les calculs sont des les filtres) Connecteurs = pipes
Avantages	Architecture flexible (interchangeable facilement) Filtres réutilisables Parallélisme possible
limites	Synchronisations entre filtre peuvent limiter le parallélisme Propagation des erreurs

Style : Layers (en couche)

Principes	Gérer différents niveaux d'abstraction au travers de couche bien identifiée
Caractérisation	Les composants communiquent avec les couches adjacentes
Avantages	Permet de décomposer les traitements (séparation des préoccupations) Réutilisation des couches
limites	Pas particulièrement rapide, redondance des traitements, localisation et traitement des erreurs difficiles, découpage difficile

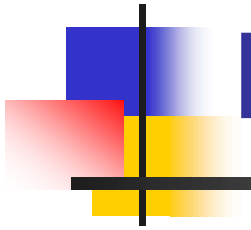
Style : N-tier / 3-tier

Principes	Décomposer par rapport à la logique métier
Caractérisation	3 couches ou plus, sur différentes machines (présentation, métier, données)
Avantages	Maintenabilité, disponibilité Passage à l'échelle Possibilité d'ajouter des couches spécifiques (exple sécurité)
limites	Plus lourd, et donc plus cher que client-serveur



Exercice 4 : Par équipe

- Chaque membre de l'équipe choisit un style parmi les suivants
  - Process-control, Repository, Blackboard
- Après avoir étudié un style, expliquez-le aux autres
- Ensemble et pour chaque style, remplissez le tableau suivant
- Répondez au quizz 4 (individuel)



En groupe e-com

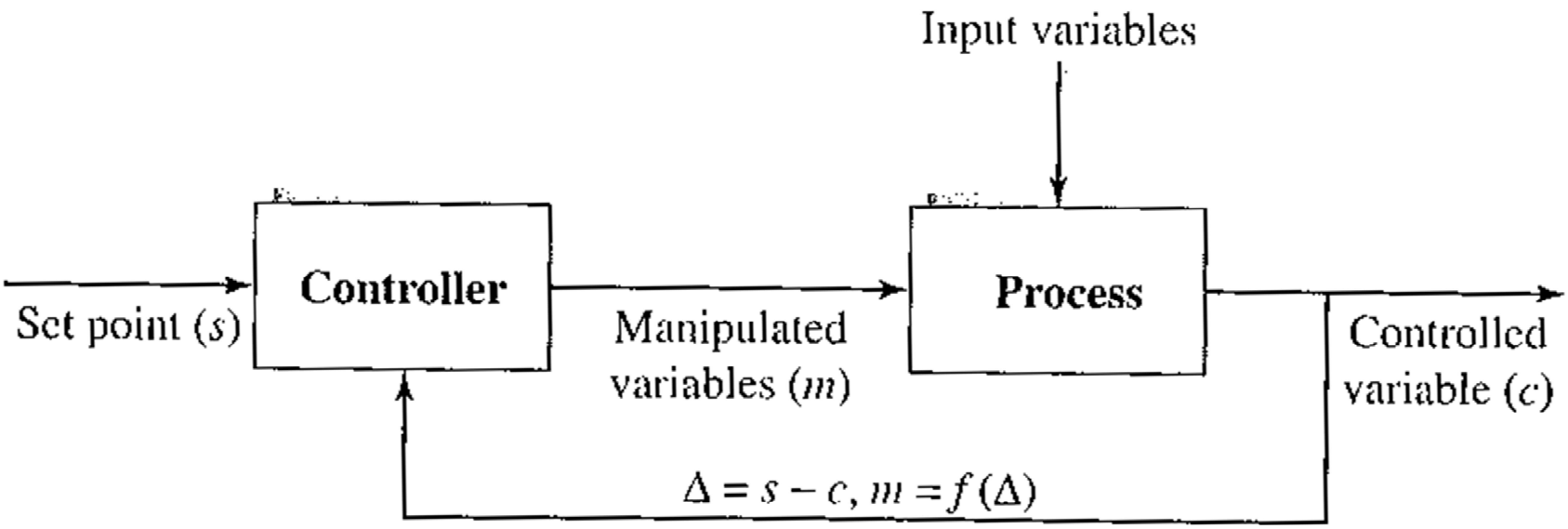
Diagramme de contexte

Liste de fonctionnalités

Style :

Principes	
Caractérisation	
Avantages	
limites	

Style : Process Control Arch.



Pour les systèmes qui doivent maintenir une sortie à un niveau donné
Pour les systèmes qui doivent atteindre un objectif fixé

Style : Process Control Arch.

Principes	Transformations des données (data-flow) Suivi de l'évolution d'une variable contrôlée
Caractérisation	2 parties Process : pour calculer les changements à faire Control : pour surveiller l'évolution Boucle de feed-back rapprochée entre les deux parties pour mesurer la différence entre objectif et état courant
Avantages	Précision du contrôle Applicable pour les systèmes embarqués
limites	

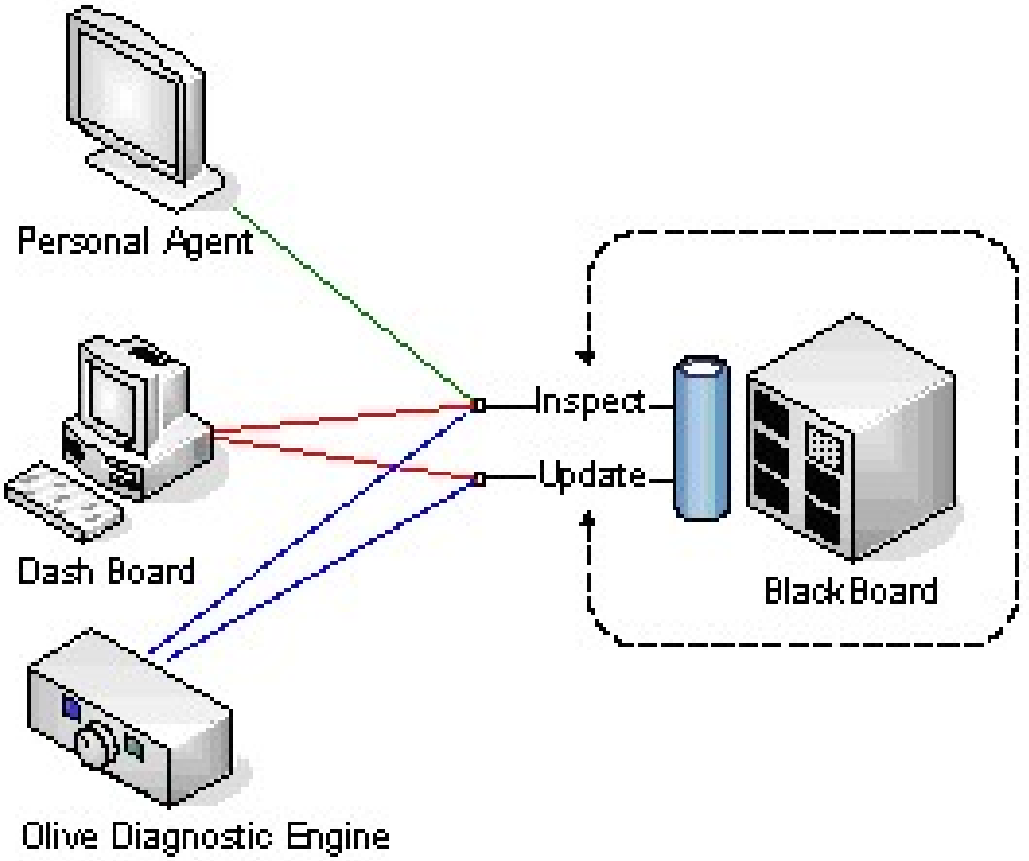
Style : Repository

Principes	Les composants accèdent et modifient les données selon les besoin des calculs
Caractérisation	L'entrepôt de données (passif) Les composants (actifs : sollicitent l'entrepôt)
Avantages	Intégrité des données : backup et restoration facile Système facile à étendre (scalability) Réutilisation des composants facile
limites	Fiabilité de l'entrepôt de données détermine la fiabilité du système Forte dépendance entre la structure de données et les composants Coût du transfert des données (systèmes distribués) Accès concurrents

Style : Blackboard

Principes	Problèmes avec des solutions non déterministes et où les composants en charge de résoudre le problème partagent des informations
Caractérisation	Des composants (sources de connaissance) Le « tableau » (contient les hypothèses et les faits) Eventuellement un contrôleur. Le tableau invoque les composants.
Avantages	Les composants peuvent être modifiés et/ou tomber en panne sans pénaliser le système Système peut être distribué, Réutilisation des sources de connaissance possible
limites	Pas de bonne solution garantie, terminaison difficile à déterminer Fort couplage vis-à-vis de la structure du tableau Synchronisation des agents vis-à-vis de l'accès aux données Difficile à valider/certifier/…

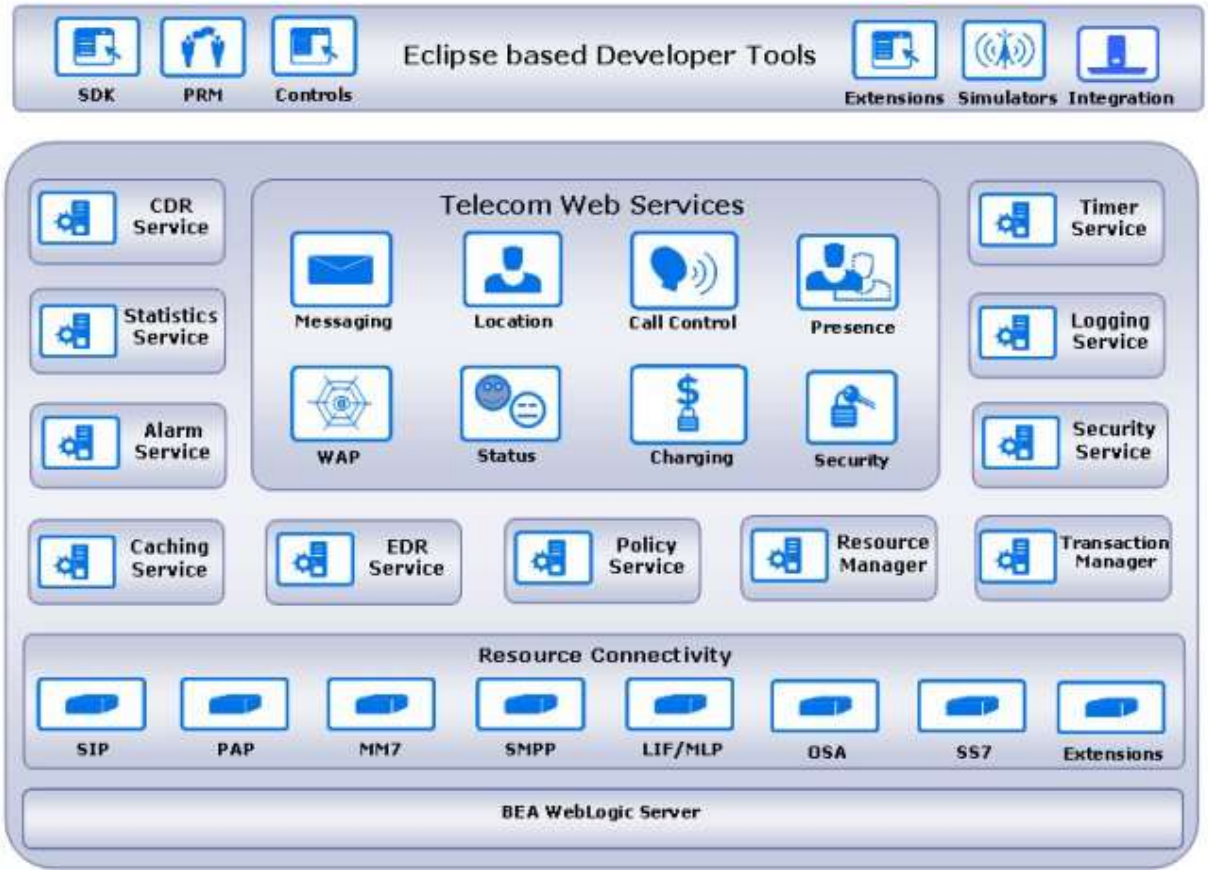
Olive Project



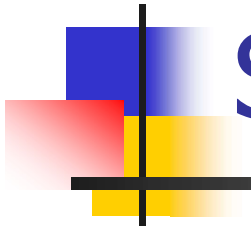
A goal is to develop a product able to make telemedicine consultations.

The system is based on **multi-agent approach**, where each expert in the Olive network is represented by his own personal agent called Dashboard (DB). DB-agents provide a formalized communications between experts, directed to solve a problem of making a diagnosis.

Traffic Paths



All traffic in Network Gatekeeper flows in traffic paths.
A traffic path consists of an application-facing interface, with Web Services Security enforcement, a Service Capability, and a network plug-in, where requests are translated between the application-facing interface and underlying network node protocols.



Cours d'architecture logicielle
Styles architecturaux

Lydie du Bousquet
Lydie.du-bousquet@imag.fr

Philippe Lalanda