Architecture Distribuée

Cours n°1

Arnaud Saval

Organisation (1)

- Cours
  - 6 x 3 heures
  - Lundi 14h00-17h00
- TPs
  - 6 x 3 heures par groupe
  - Vendredi 14h00-17h00
- Notation
  - une note par TP (code source + rapports)
  - un examen (questions + exercices)

Organisation (2)

- Intervenants
  - Arnaud Saval

    Ingénieur de recherche Airbus Defence & Space

    arnaud.saval@gmail.com

  - Esther Nicart

    Ingénieur de recherche Airbus Defence & Space

    esther.nicart@gmail.com

  - Arthur Vaisse-Levesten

    Ingénieur de recherche Airbus Defence & Space

    arthurvaisse@yahoo.fr

Objectifs du cours “Architectures distribuées”

- Compréhension des motivations
- Conception d'une architecture distribuée
- Maîtrise des principaux modèles
- Aperçu des problèmes posés
- Introduction aux principaux frameworks existants

Aperçu du cours

- Introduction
- Problème de conception d'architecture
- Architecture logique & matérielle
- Système distribué
- Modèles d'architecture
  - Client/Serveur
  - 3-tiers
  - N-tiers
  - Pair à pair (P2P)
  - Virtualisation

Quelques références

- Software Architecture: IEEE Standard 1471-2000
- P. Kruchten, Architectural Blueprints—The “4+1” View Model of Software Architecture, IEEE Software 12 (6), Nov. 1995, pp42-50
- Tanenbaum & van Steen, Distributed Systems, Principles and Paradigms, seconde édition
- Architecture of Distributed Systems, cours de Johan Lukkien, 2011
- Architectural Patterns Revisited – A Pattern Language, Paris Avgeriou & Uwe Zdun, 2005
- Software Architecture, Foundations, Theory, and Practice, R.N. Taylor, N. Medvidovic, E.M. Dashofy, Wiley & Sons, 2009
- Software Architecture in Practice, Second Edition, L. Bass, P. Clements, R. Kazman, SEI Series in Software Engineering, Addison-Wesley, 2003

INTRODUCTION

Quelques questions

- Qu'est-ce qu'une
  - Architecture système ?
  - Architecture logicielle ?
- Quelles sont les différences entre architecture et conception ?
- Comment identifier l'architecture d'un système existant ?
- L'architecture est-elle une documentation ou une spécification ?

Architecture logicielle... WTF !

- A partir de quand est nécessaire une architecture ?
  - Pour un site web ?
  - Pour une application ACCESS ?
  - Pour une traitement sur 3... 12... 200 serveurs ?
- Quelques critères pour définir si il est nécessaire de mettre en place/concevoir une architecture :
  - Le système est-il complexe ?
  - Y-a-t'il plusieurs alternatives de conception ?
  - Dois-je partager le travail avec une équipe ?
  - Doit-on découper le travail ?
  - Aura-t-il besoin d'être maintenu ?

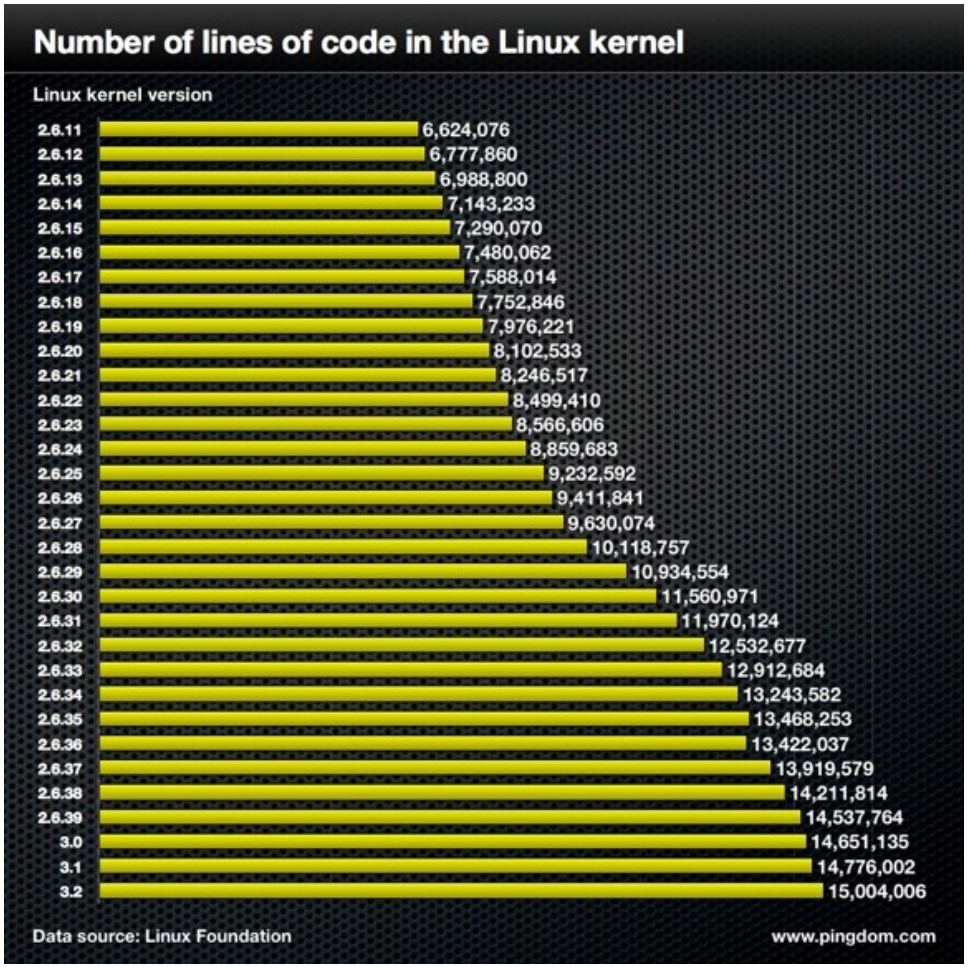
Architecture logicielle...

La plupart des logiciels actuels sont devenus trop complexes pour être maintenu !

Kernel Linux 1.0.0 – 1994

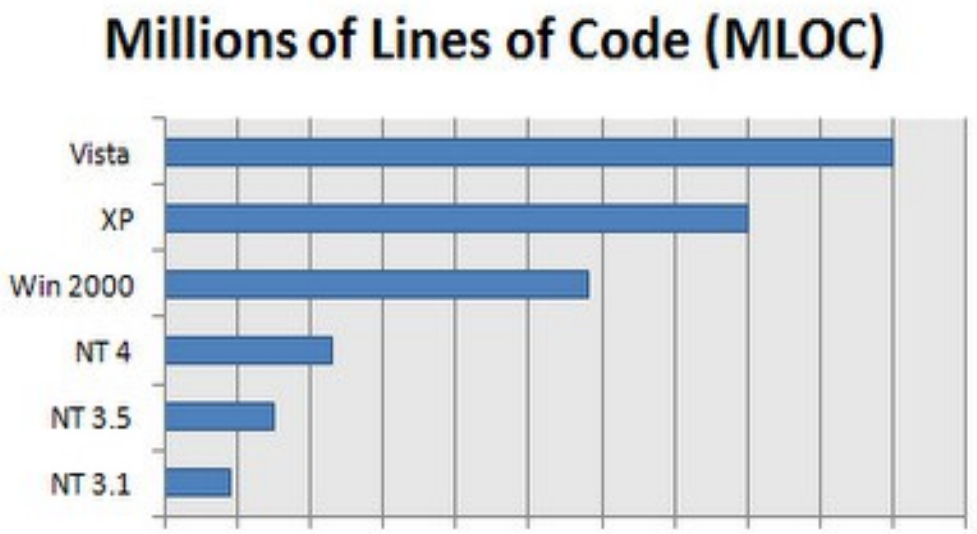
176250 Lignes
de codes (LOC)

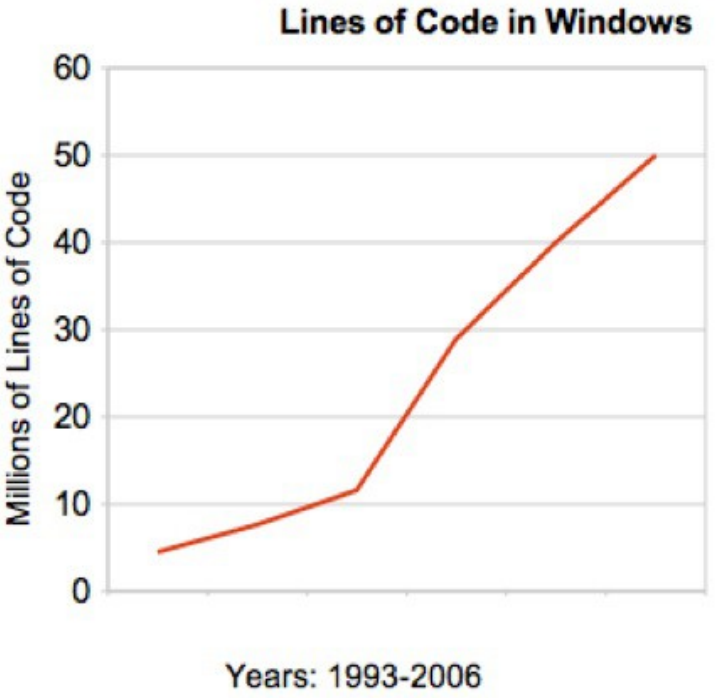
Aujourd'hui : plus de
16M LOC



Architecture logicielle...

La plupart des logiciels actuels sont devenus trop complexes pour être maintenu !





Architecture logicielle...

- Le logiciel qui gère un Boeing 787 fait environ 7M de lignes de code (LOC) – 3 X celui du 777
- Logiciel de l'airbus A3XX : plus de 10M
- Les voitures contiennent un système embarqué avec plus de 10M LOC
- Le futur système combat US est estimé à 60M LOC

$\longrightarrow$ Si le problème est trop complexe : on le découpe

Remarques générales (1)

- Analyse des problèmes
  - Besoin de comparer des solutions potentielles
  - Les performances systèmes globales doivent être estimées
  - Il doit être possible de prédire les comportements du système dans des cas complexes
- Communication
  - Les grands systèmes sont fait en équipe (distribuées ?)
  - Chaque équipe travaille à des niveaux différents
  - Chaque personne à une culture & un historique propre

Remarques générales (2)

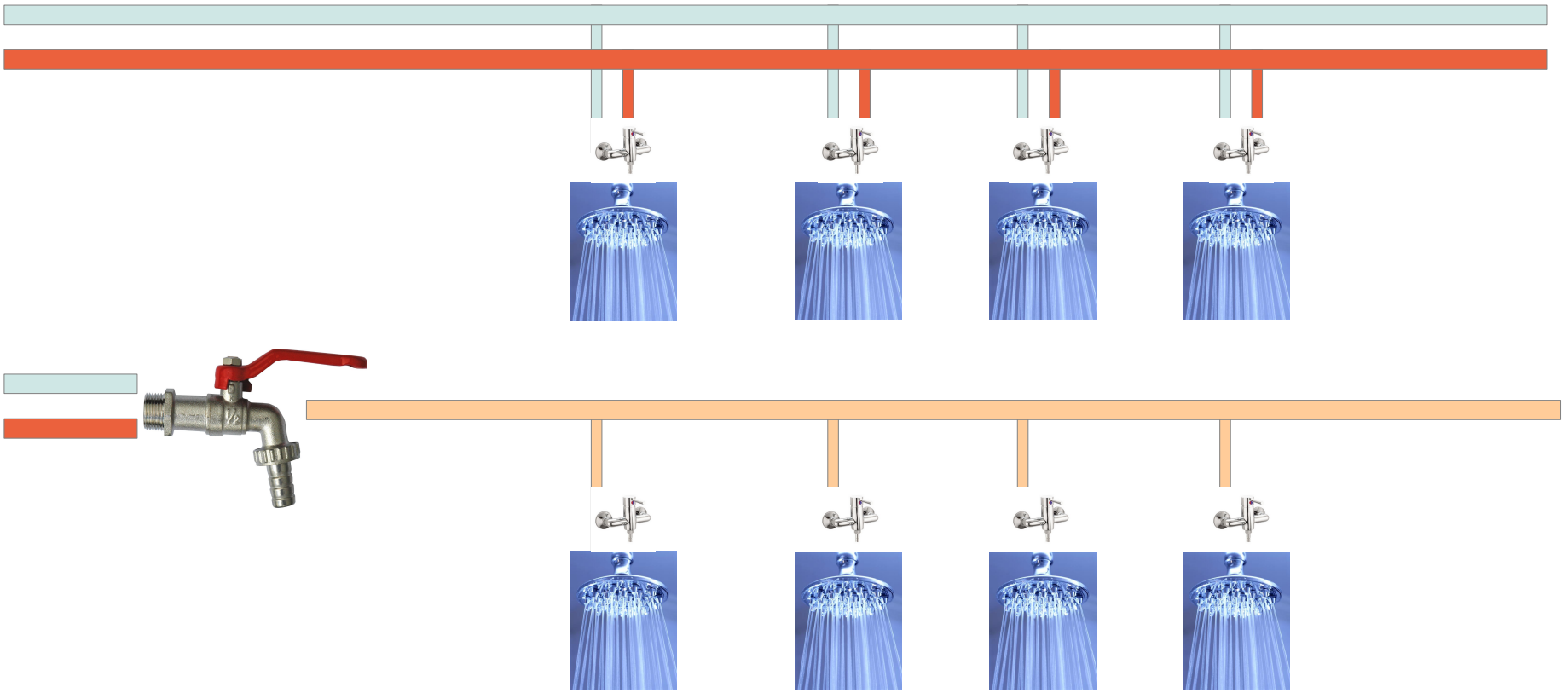
- Construction
  - Le processus de développement n'est pas “one-shot”
  - Les sytèmes sont souvent des assemblages de briques existantes
- Compréhension/documentation
  - La structure d'un système n'a pas besoin de détails pour être comprise
  - Une personne “nouvelle” doit pouvoir comprendre le système même après sa mise en oeuvre
  - La communication avec les utilisateurs et les clients doit être faite dans les temps et simplement

PROBLEME DE CONCEPTION

Exemple : système de douches

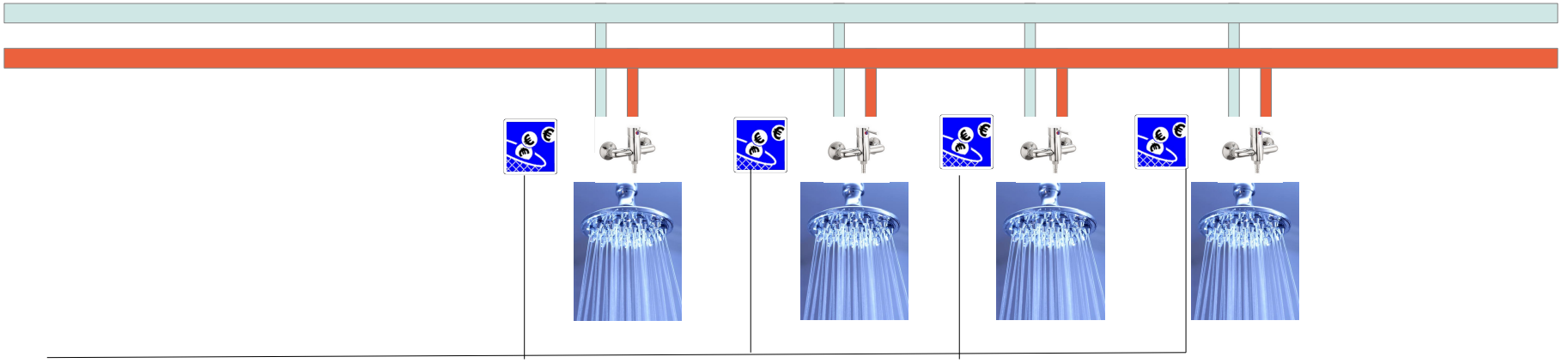
Exemple : système de douches

Deux solutions alternatives :



Exemple : système de douches

Ajout d'un système de paiement :



Besoin d'une infrastructure supplémentaire

Système local complexe :

robinet froid/chaud + système de paiement

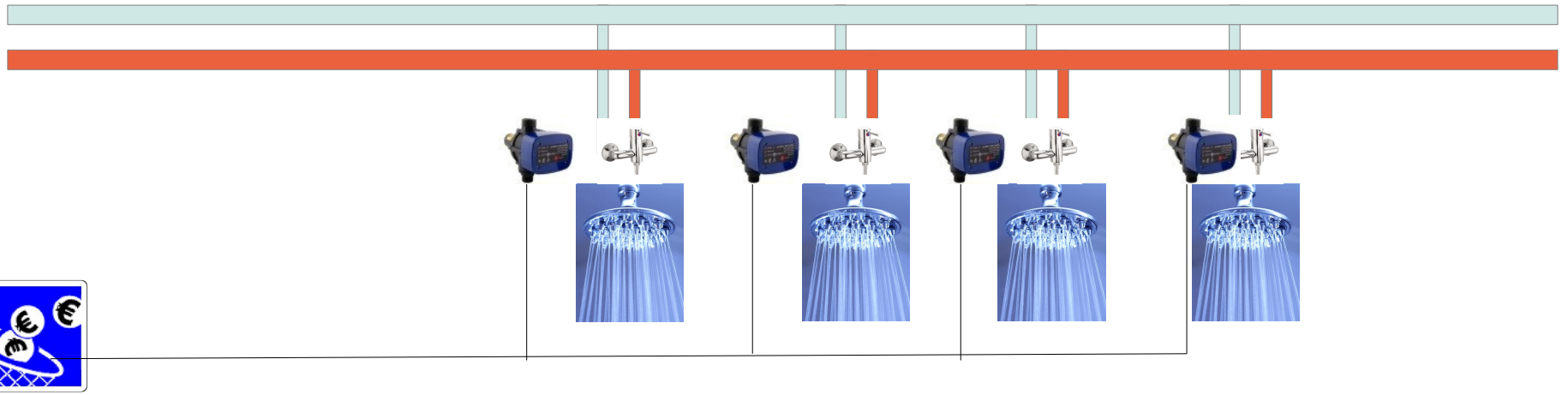
Alternative ?

Exemple : système de douches

Un système centralisé !

Exemple : système de douches

Système de paiement centralisé...



Collecte d'argent centralisée

infrastructure allégé : robinet contrôlé

Rend les douches longues difficiles...

Problème de conception d'architecture : les douches (1)

- Besoins fonctionnels :
  - Utilisation simple des douches - cas d'utilisation primaire
  - Entretien & maintenance - cas d'utilisation secondaire
- Eléments du système :
  - Douche / Robinet
  - Système de contrôle
- Connecteurs :
  - Tuyauterie

Problème de conception d'architecture : les douches (2)

- Contraintes :
  - Choix des éléments (coût, qualité, entretien, lois...)
- Environement technique :
  - Bâtiment,
  - Tuyauterie existante, capacité de la chaufferie
  - …
- Paramètres non-fonctionnels:
  - Performance (litre/seconde ; litre/douche)
  - ...

Problème de conception d'architecture : les douches (3)

- Hypothèses de travail
  - Douche fréquence matin/soir
  - Tendance à une utilisation trop longue
- Parties prenantes
  - Utilisateurs des douches
  - Agents d'entretien
  - Installateurs
  - Réparateurs
  - …

$\longrightarrow$ chaque partie à une fonction précise et une vision précise vis-à-vis du système

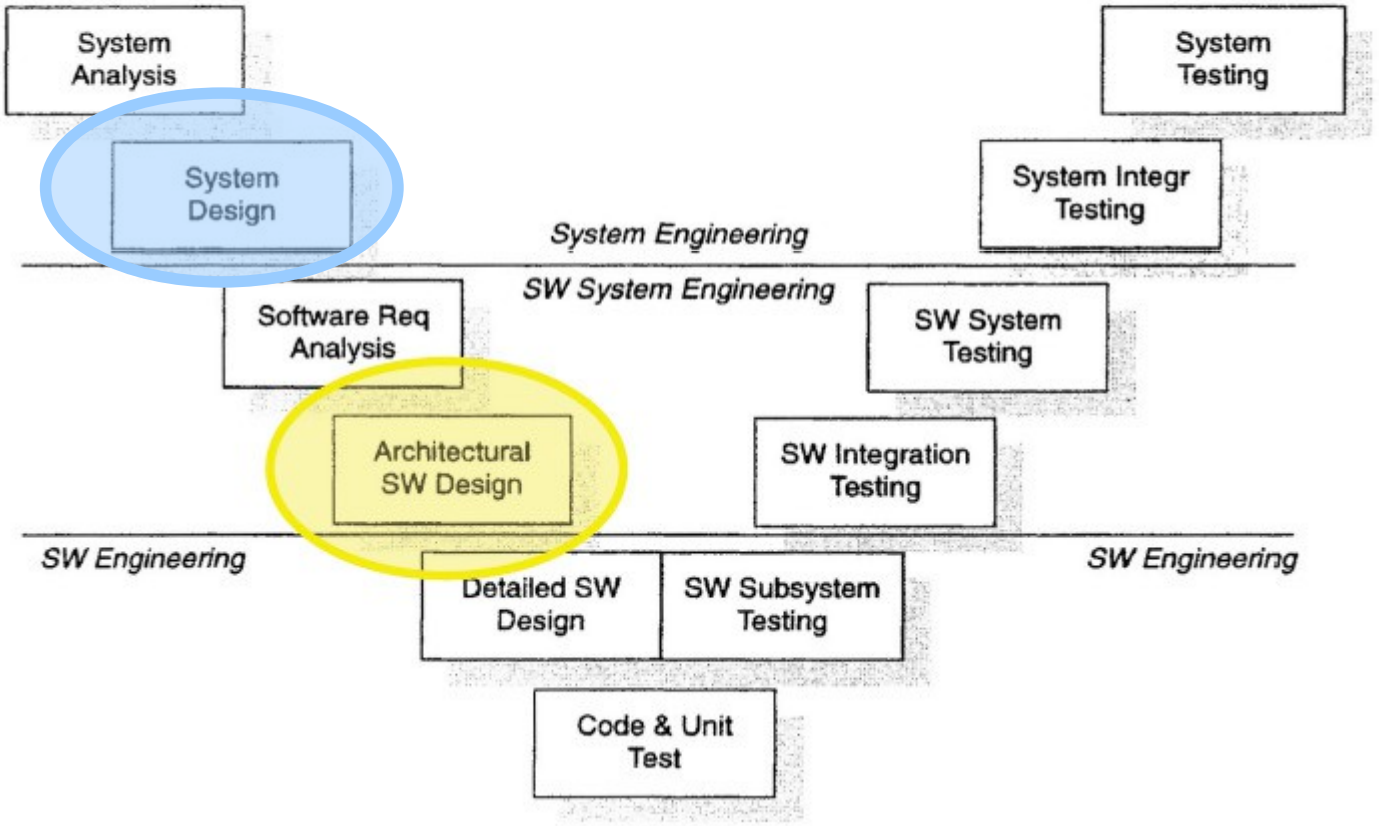
Conception d'architecture

Conversion d'un problème en (un modèle) de solution

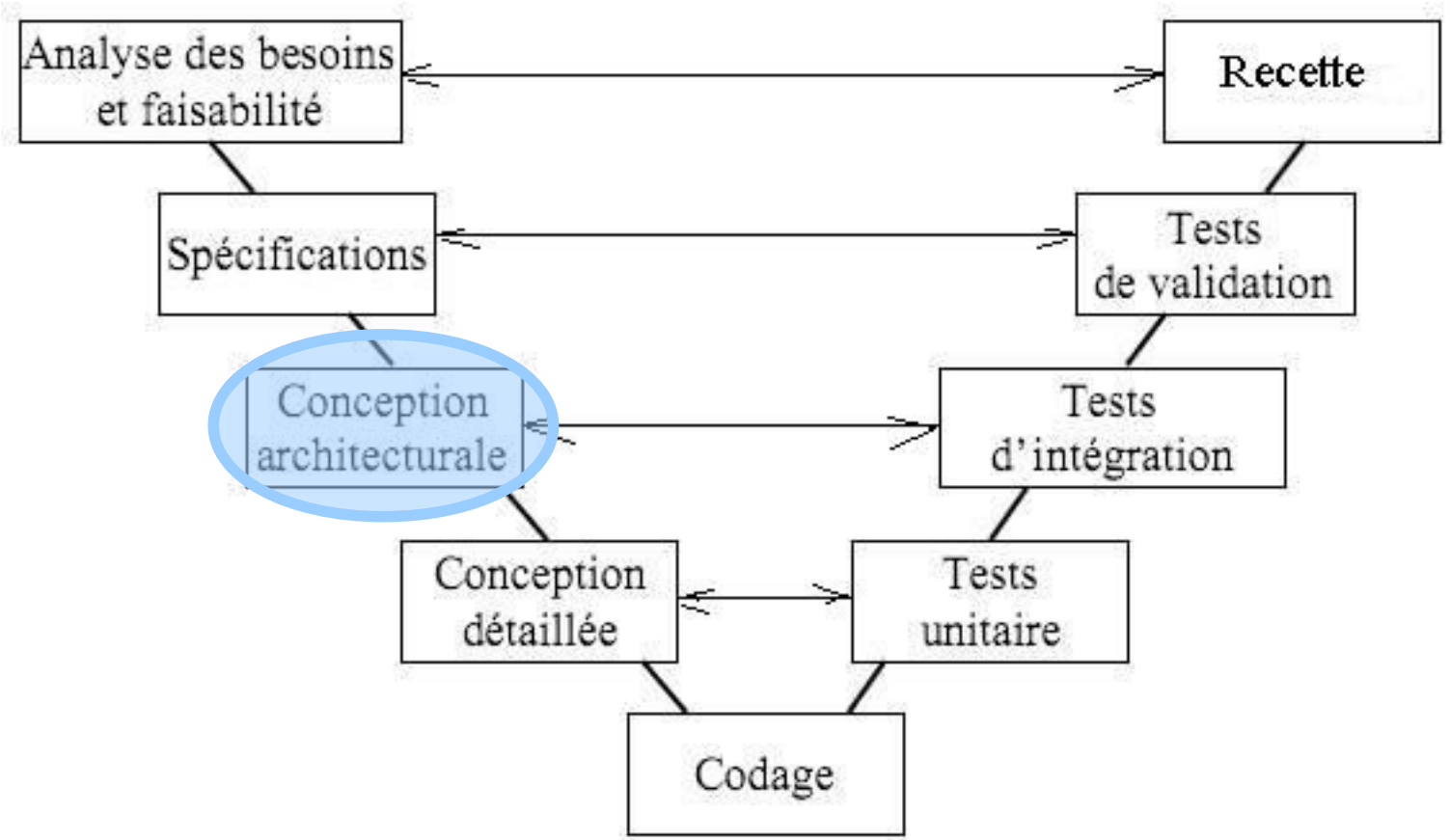
- Besoin d'analyser les beoins/enjeux/contraintes
- S'adapter à l'environement
- Visualiser la vie du système
- Identifier et faire participer TOUTES les parties prenantes

Conception d'architecture dans le développement d'un système

From: M.J. Christensen, R.H. Thayer. *The Project Manager's Guide to Software Engineering's Best Practices*. Wiley, 2002



Conception d'architecture dans le développement d'un logiciel



Cycle en V du développement logiciel

Conception d'architecture : un processus

Conversion d'un problème en (un modèle) de solution

- Processus itératif :
  - Chaque niveau de conception N+1 solutionne les problèmes du niveau N
  - Les modèles sont de plus en plus détaillés jusqu'à ce qu'ils ne comportent plus de “problèmes” à résoudre
- Processus hiérarchique :
  - Les éléments de chaque niveau sont différents
  - Spécialisation des intervenants

Conception d'architecture : un processus

4 domaines travail :

- Analyse du domaine
  - Cas d'usage
  - Echange avec les parties prenantes
- Mise en oeuvre de méthodes de conception
  - Décomposition hiérachique
  - Design patterns
- Synthèse
  - Evaluation et sélection des alternatives de conception
- Validation
  - Le système répont-il au besoin ? Ça compile ? Ça tourne ?

ARCHITECTURE LOGIQUE & MATERIELLE

Architecture logique (1)

"The fundamental organization of a system embodied by its **components**, their **relationships** to each other and to the environment and the **principles guiding its design** and evolution"

IEEE Standard P1471 Recommended Practice for Architectural Description of Software-Intensive Systems

Architecture logique (2)

Un paquet de modèles/schémas

- organisés en “vues”
- à différent niveaux d'abstraction
- le niveau “0” correspond au schéma d'architecture principal

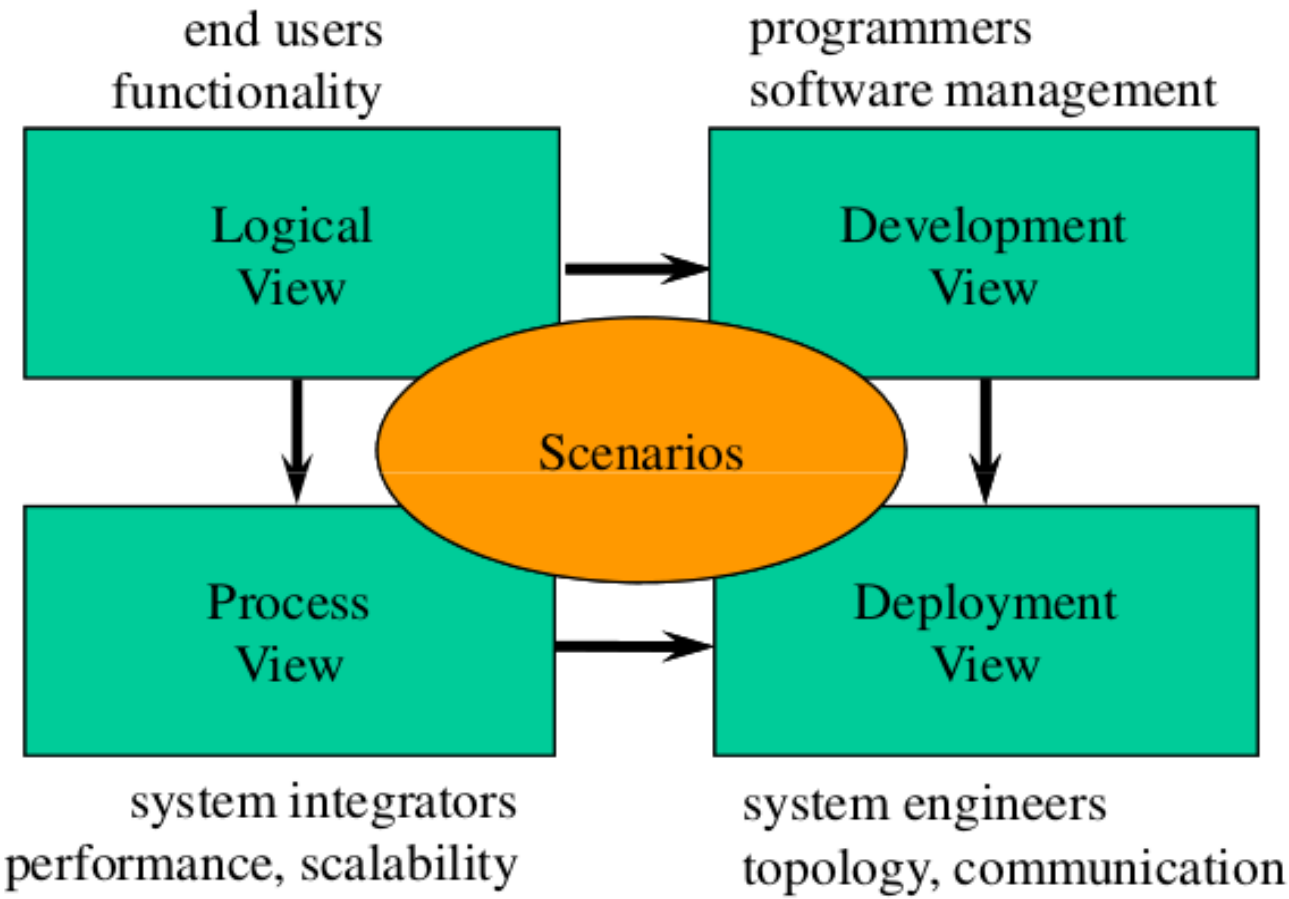
Un modèle (d'un système) = abstraction des composants et communication/comportements

Architecture logique (3)

- Un modèle (d'un système) = abstraction des composants et communication/comportements
  - représentation sans détails non pertinants
  - préserve unqiuement les propriétés importantes à l'échelle d'analyse considérée

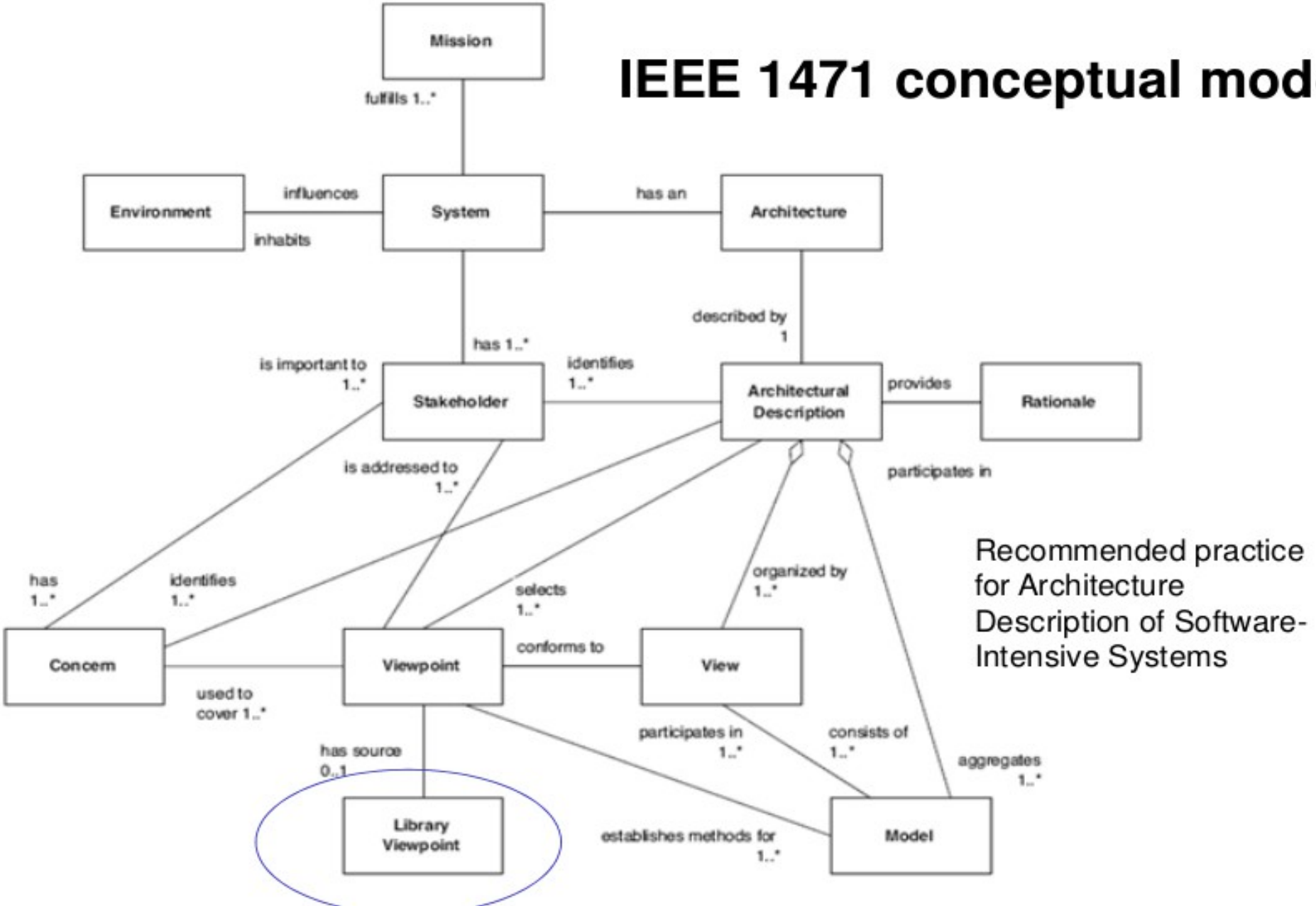
- Une vue = une façon d'observer et de décrire le système selon le point de vue d'une des parties prenantes
  - formelle ou informelle, dessin, interview...
  - décrit les éléments d'intérêts pour une des parties prenantes

Modèles de vues (1)

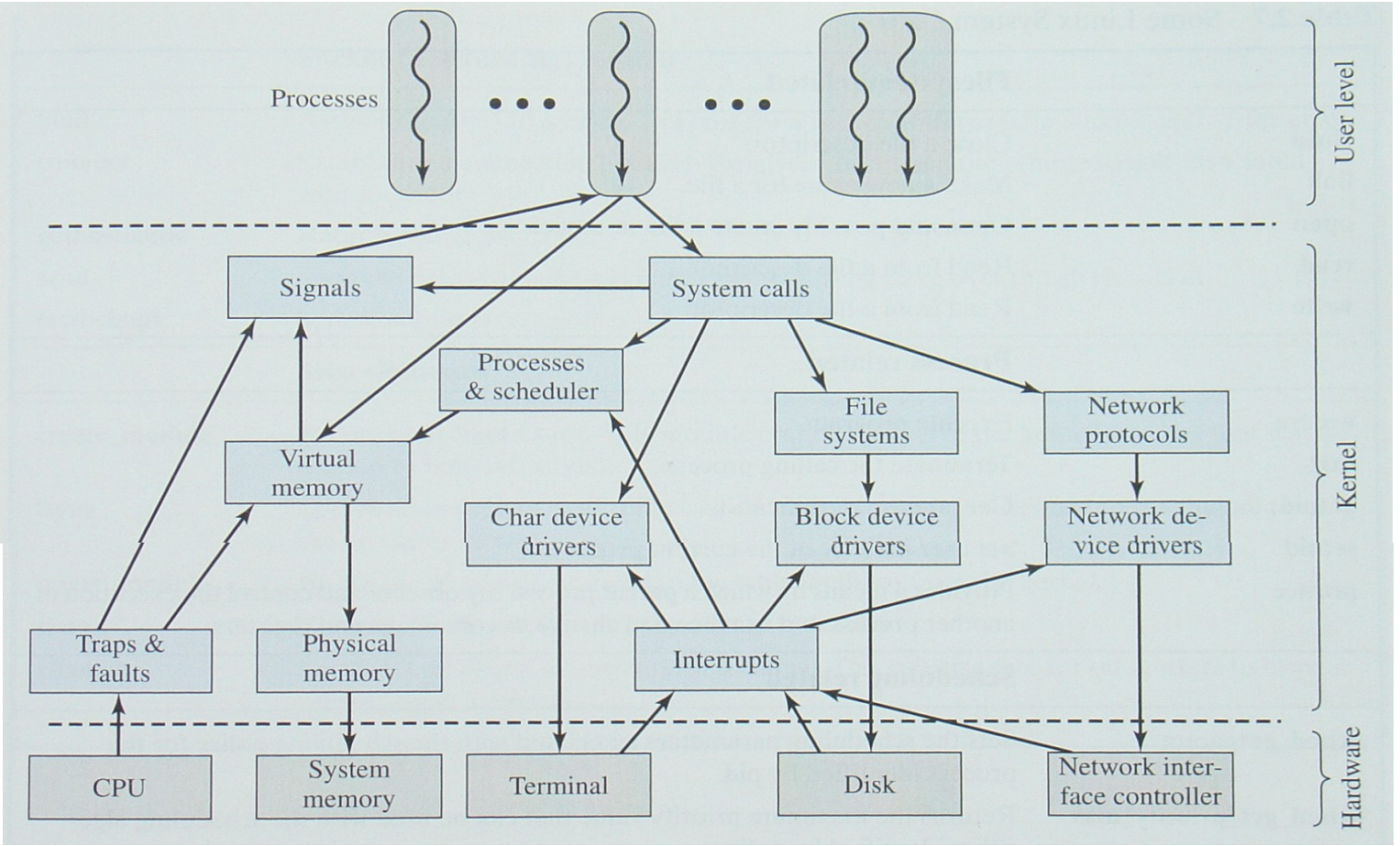


4+1 View Model [Kruchten 95]

Modèles de vues (2)

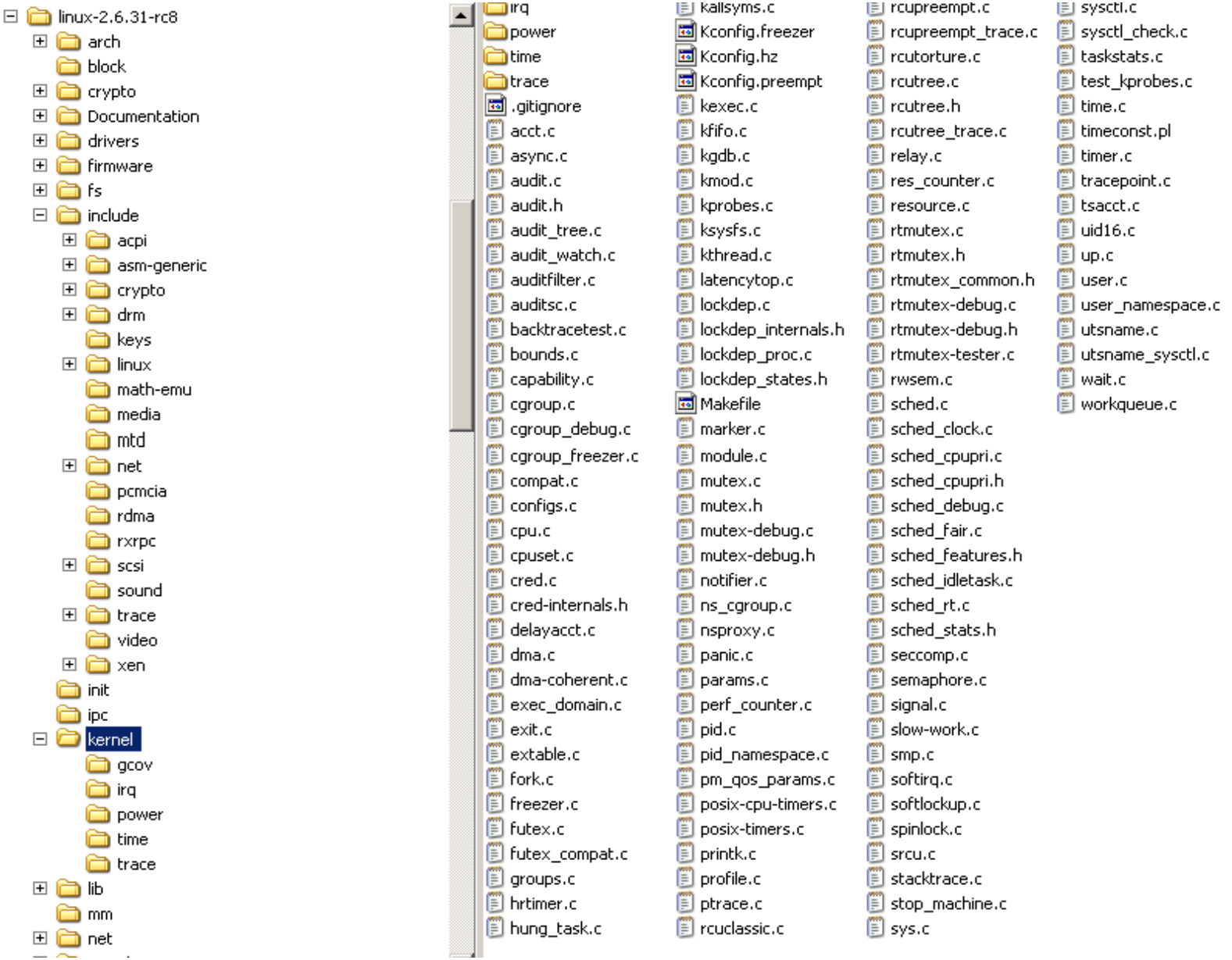


Exemple de vue logique : unité fonctionnelle du kernel Linux

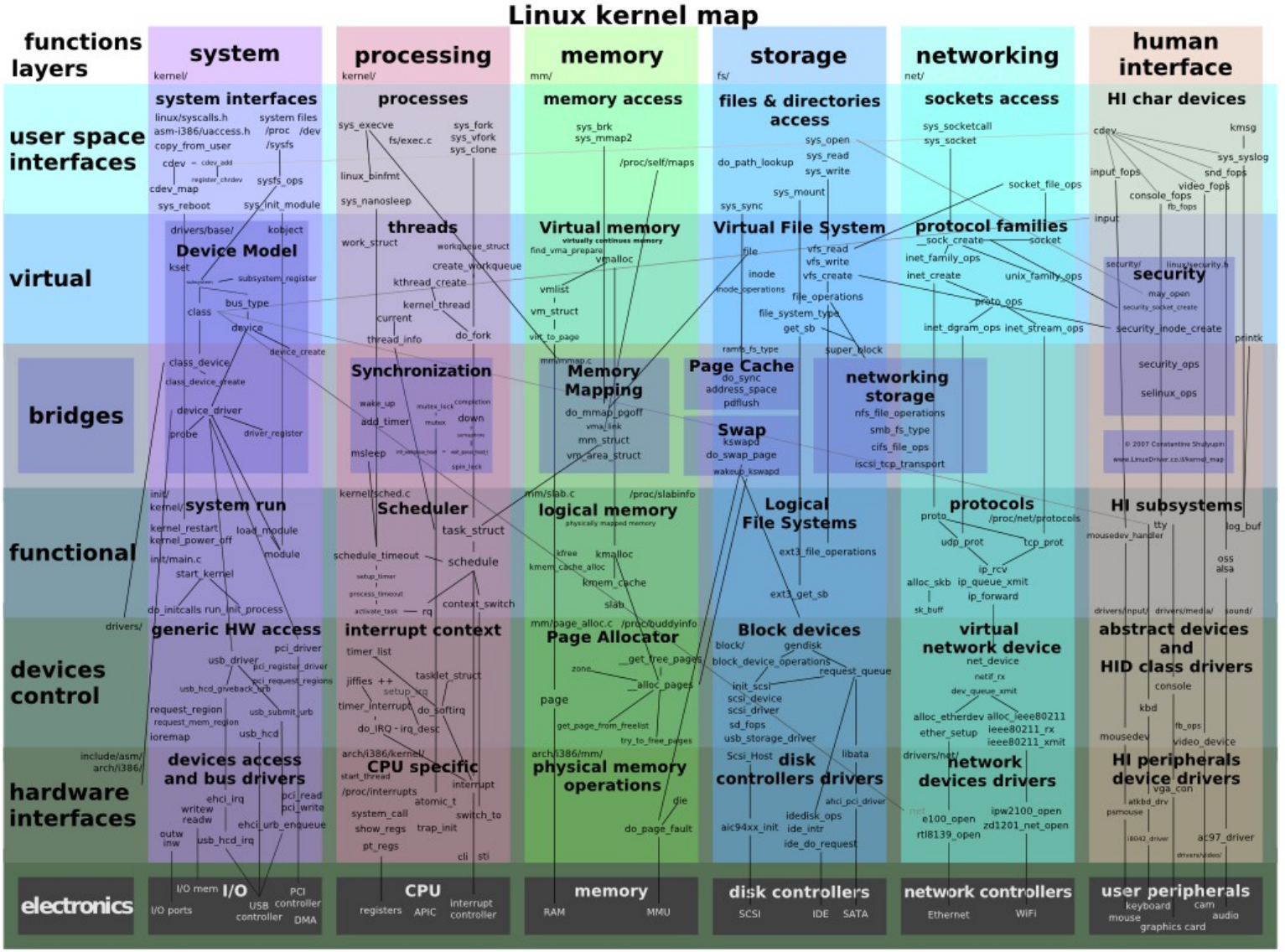


Interprétation ambigüe : que représentent les boîtes ? Les axes ?

Exemple de vue : sources



Exemple de vue: Vue cartographique



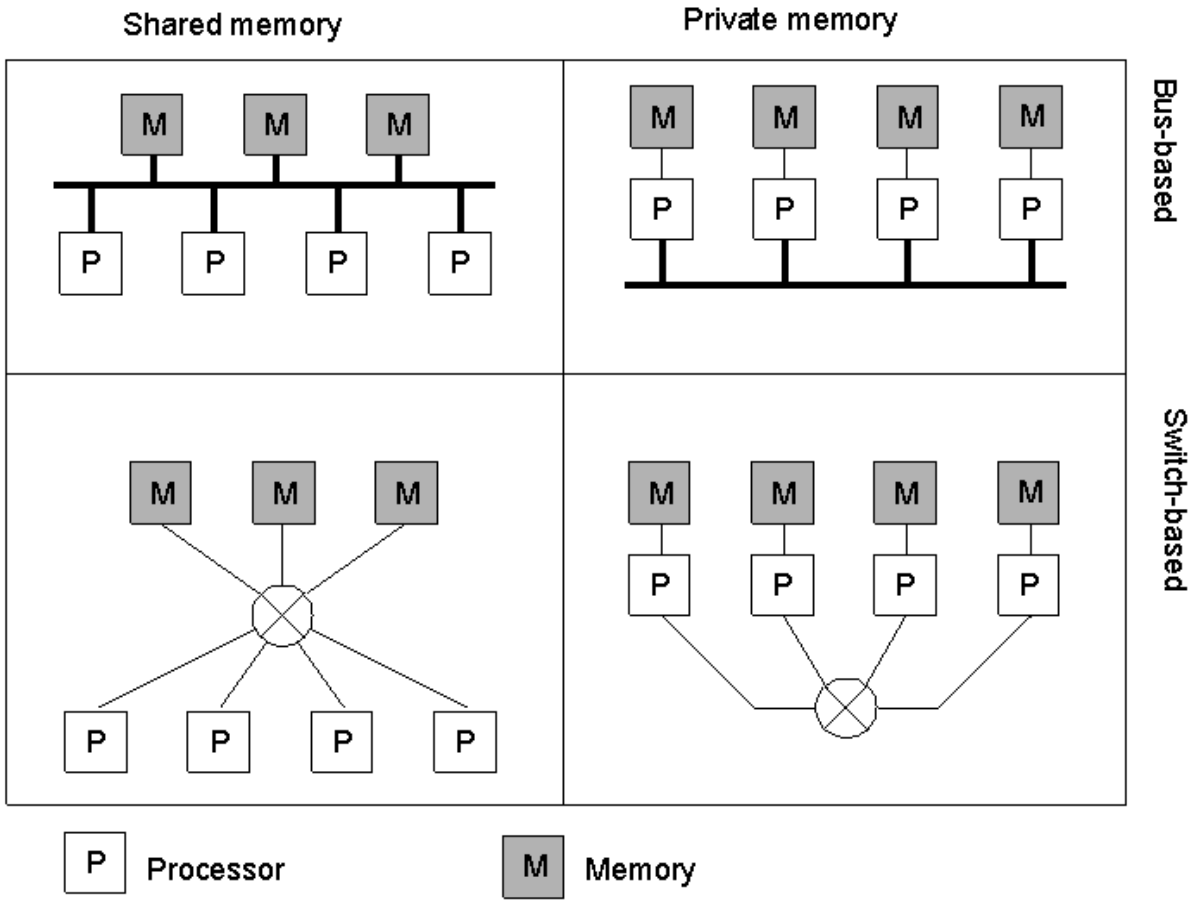
Source : http://www.makelinux.net/kernel_map

Questions à se poser

- Quels sont mes éléments fonctionnels (à ce niveau), mes connecteurs, mes contraintes?
- Comment s'articulent les schémas entre eux ?
- L'architecture couvre-t-elle tous les problèmes à résoudre ?
- Comment mon architecture va-t-elle permettre de sélectionner une alternative ? Sert-elle les décisions à prendre?
- Comment y intégrer des nouvelles fonctionnalités ?
- Quels sont les éléments posés/fixés par l'environement ?
- Quelles les capacités système importantes/indispensables/trop chères ?
- **L'architecture proposée est-elle réalisable ?**

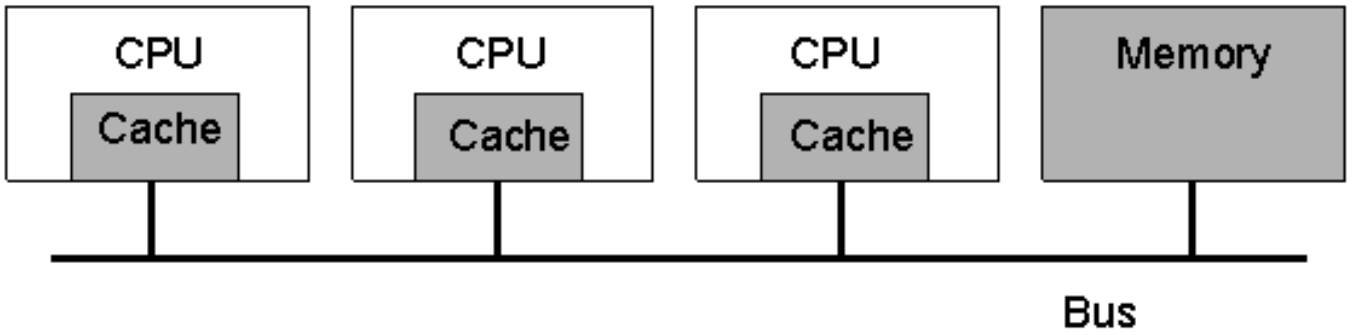
Architecture physique

Organisation des éléments CPU & mémoire



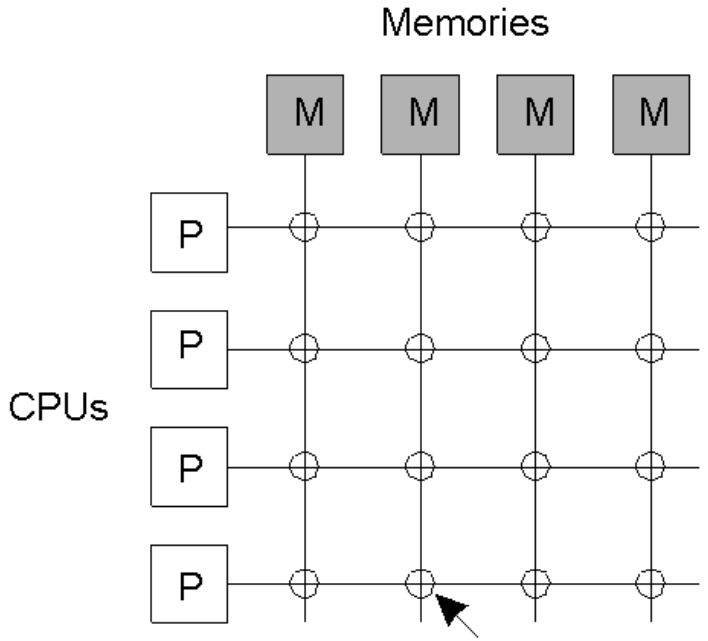
Architecture multi-processeur (1)

- Basée sur un bus d'échange commun et une mémoire partagée

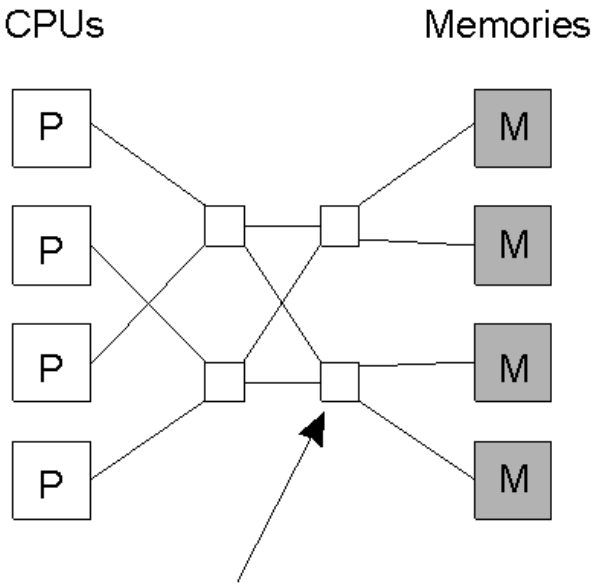


Architecture multi-processeur (2)

- Basée un échangeur en grille ou en réseau



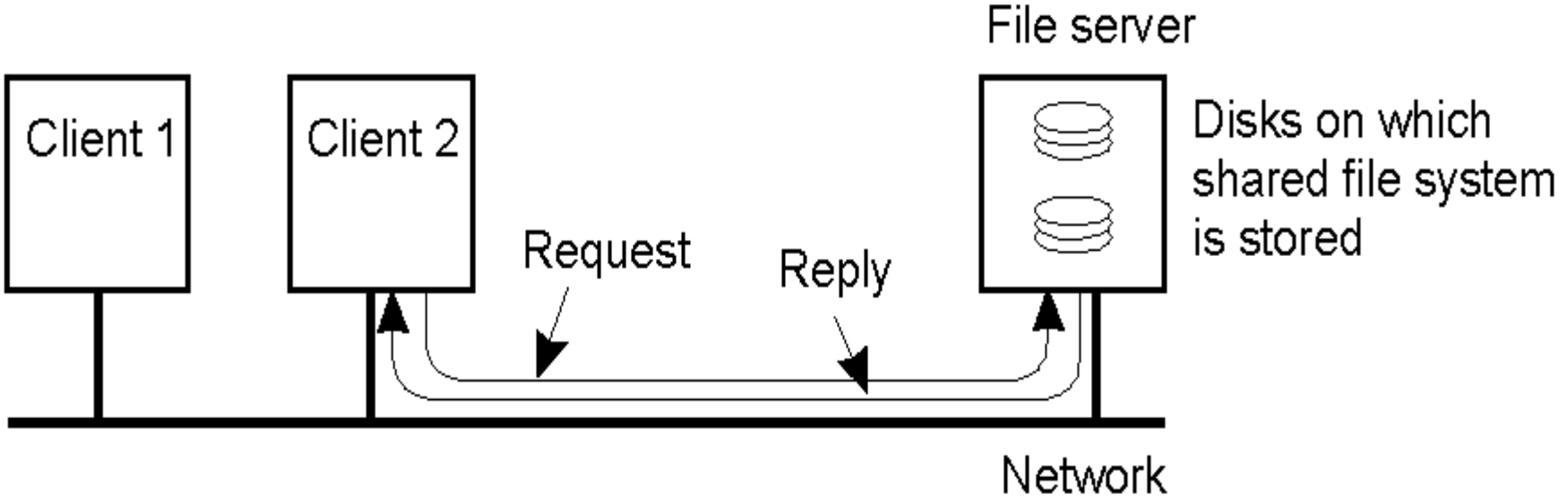
(a)



(b)

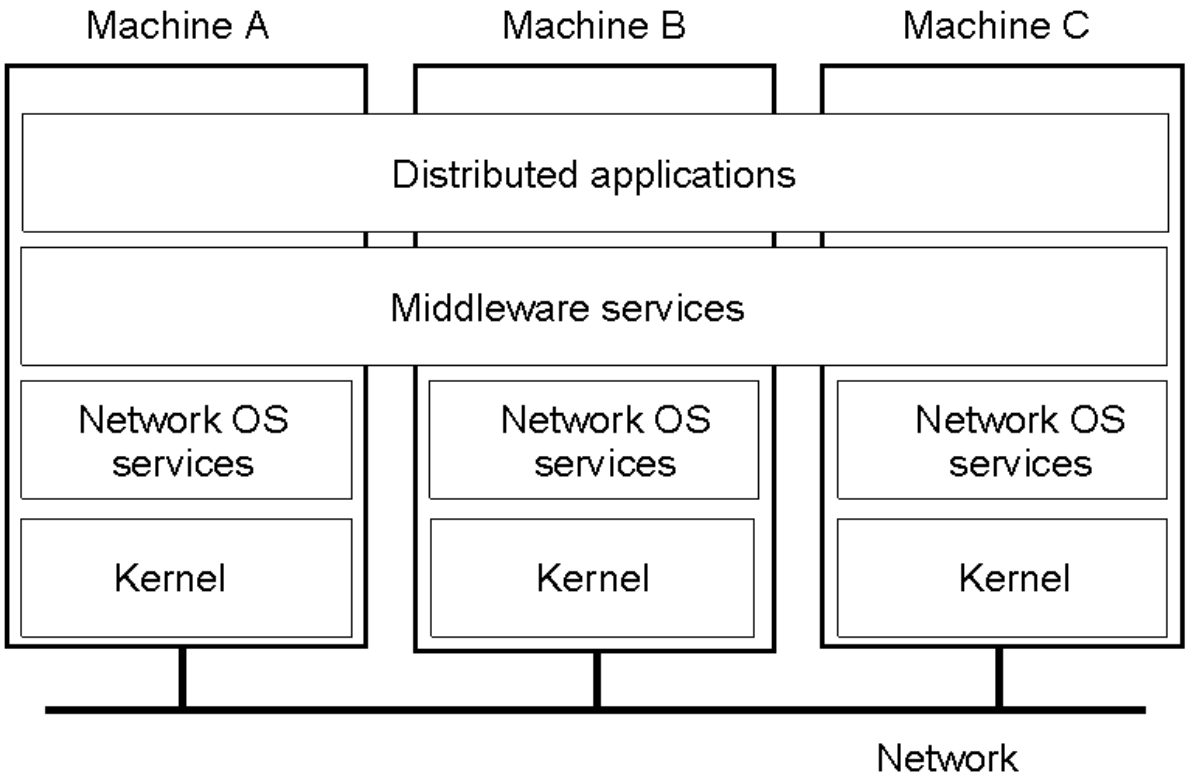
Architecture multi-serveurs (1)

Distribution d'un système de fichier



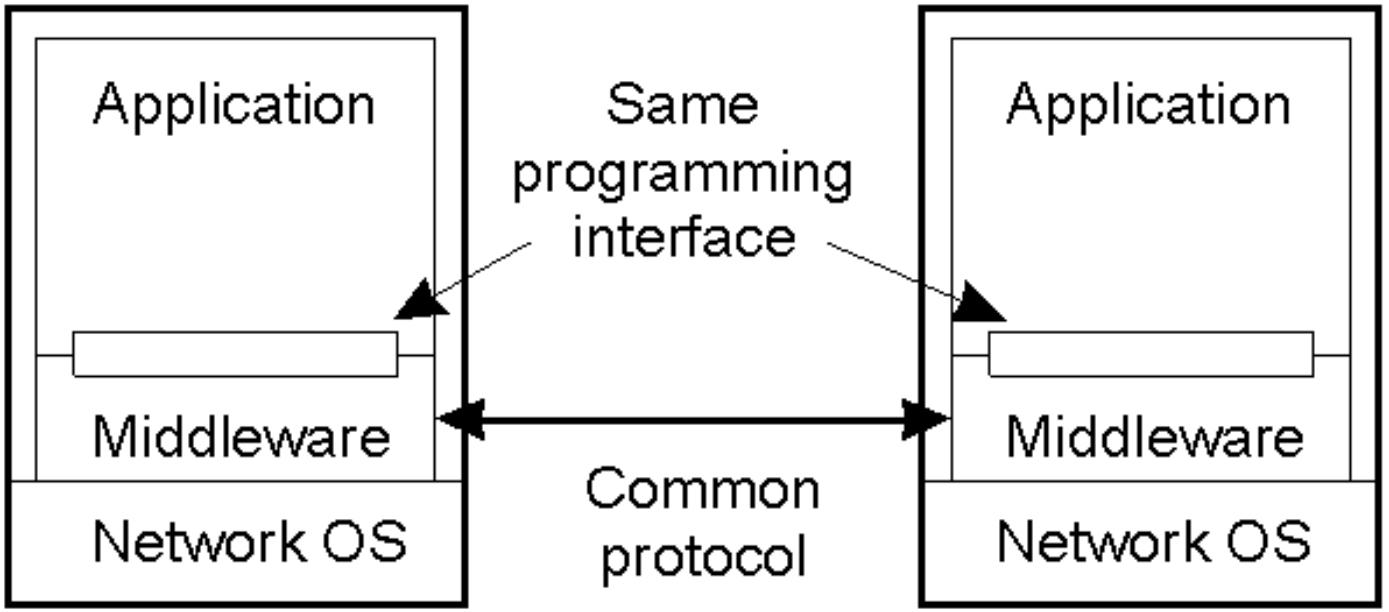
Architecture multi-serveurs (2)

Distribution d'applications et middleware



Architecture multi-serveurs (3)

Si un middleware est utilisé, une interface (API) et un protocole communs doivent être choisis



Références (one more time)

- Software Architecture: IEEE Standard 1471-2000
- P. Kruchten, Architectural Blueprints—The “4+1” View Model of Software Architecture, IEEE Software 12 (6), Nov. 1995, pp42-50
- Tanenbaum & van Steen, Distributed Systems, Principles and Paradigms, seconde édition
- Architecture of Distributed Systems, cours de Johan Lukkien, 2011
- Architectural Patterns Revisited – A Pattern Language, Paris Avgeriou & Uwe Zdun, 2005
- Software Architecture, Foundations, Theory, and Practice, R.N. Taylor, N. Medvidovic, E.M. Dashofy, Wiley & Sons, 2009
- Software Architecture in Practice, Second Edition, L. Bass, P. Clements, R. Kazman, SEI Series in Software Engineering, Addison-Wesley, 2003