IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Interaction et Vérification

A. Beugnard

IV – C1

2017

- La notion d'interaction
- La notion d'interface
- Un peu d'architecture
- Des modèles d'interaction
- Contractualiser les interactions
- Des diagrammes d'interaction

- Entre machine et humain (IHM)
- Entre machines (réseaux, systèmes distribués)
- Entre application, programmes
- Entre agents (logiciels)
- Entre humains

Comment décrire/spécifier une interaction ?

Après, on peut faire des vérifications. . .

A problem well stated is on its way to solution

Bergson, XXth

concurrence Plusieurs activités, communication fiable, sûre, existence d'une horloge commune

distribution Communication lente, non sûre, pas d'horloge commune, état global à construire

point de vue/aspect	optimiste	pessimiste
concurrence	vitesse (load-balancing)	entrelacement (incorrection)
distribution	résistance aux fautes (haute disponibilité)	fautes (incorrection)

Au cœur des problèmes : les interactions !

Agenda

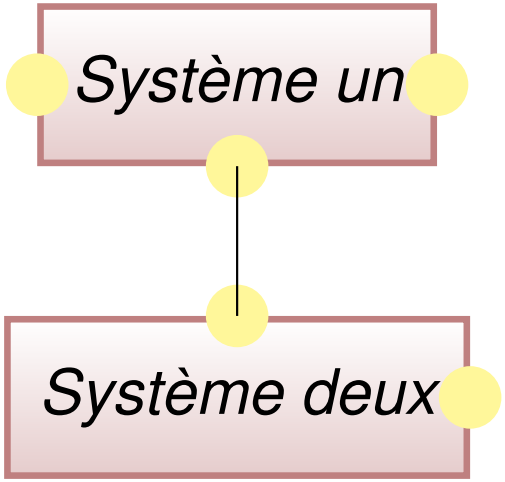
- 2h assez abstraites
- 2h de modèles divers
- 2h de diagrammes

- Voir la bibliographie
- Cours de Bernard Espinasse (Univ. Aix-Marseille) : Communication et langages de communication dans les SMA
- Cours de Rachid Guerraoui (EPFL) : Distributed Algorithms

Prolégomènes

Action ou influence réciproque qui peut s'établir entre deux objets (ou personnes) ou plus. Une interaction se décompose en plusieurs séquences, échanges et tours de parole.

Wikitionnaire



- On dispose d'au moins 2 objets/acteurs/systèmes
- Ils échangent (des informations, de la matière, de l'énergie, . . .)

Doit-on décrire le système complet ? ou chacun des objets/acteurs/systèmes ?

- *Interaction* dans le premier cas
- *Interface* dans le second

Une *interaction* décrit les échanges entre des systèmes.
Les systèmes offrent des *interfaces*.
Les interfaces décrivent les règles, les hypothèses, le *contexte* (supposé, légitime) de l'interaction.

Choisir un langage de description d'interface (ou d'interaction), c'est choisir des règles, des contraintes, des propriétés. . .

Interface

- Qu'est-ce qu'une interface ?
- Qu'est-ce qu'un système ?
- Pourquoi s'y intéresser ?
  - Courant
  - Utile (abstrait)
  - Sans, ou mal définies, génère de gros soucis pendant l'integration, le test voire en service

Des photos ? eau/air ; cellule ; cables ; chip ; API ; mécanique - vue éclatée ; IHM ; téléphone (plusieurs niveaux : H-H et S-S)

Des soucis fameux

- Retard de l'A380
- Satellite US/EU
- Couplage de 2 rames de TGV

Dû à la complexité du câblage : 530km, 100000 cables et 40300 connecteurs.
Source : Cadalyst magazine article, *What Grounded the Airbus A380 ?* 6 Décembre 2006 By : Kenneth Wong [1]
Information personnelle, sources officieuses : les concepteurs Allemands et Français n'utilisaient pas la même version du logiciel Katia ce qui a provoqué des erreurs d'alignement.

1. http://www.cadalyst.com/management/what-grounded-airbus-a380-5955

Point à la frontière entre deux éléments, par lequel ont lieu des échanges et des interactions
Exemples

- Humain - Humain
- Humain - Ordinateur/Système (IHM)
- Système - Système

- À quoi ça sert ?
  - À contrôler un système
  - À interagir (demander un service, récupérer ou donner une information) avec un système
- Comment ça se décrit ?
  - Un schéma, un plan, un mode d'emploi, un contrat, etc.
- Comment ça s'utilise ?

But d'une interface

- Assurer le couplage
- Assurer l'interopérabilité
- Mettre les 2 parties d'accord
  - Type de donnée, unité, ordre des messages (protocole), intensité, etc.

- Multiplier les couplages
- Réduire le couplage
  - Médiateur
  - Pivot

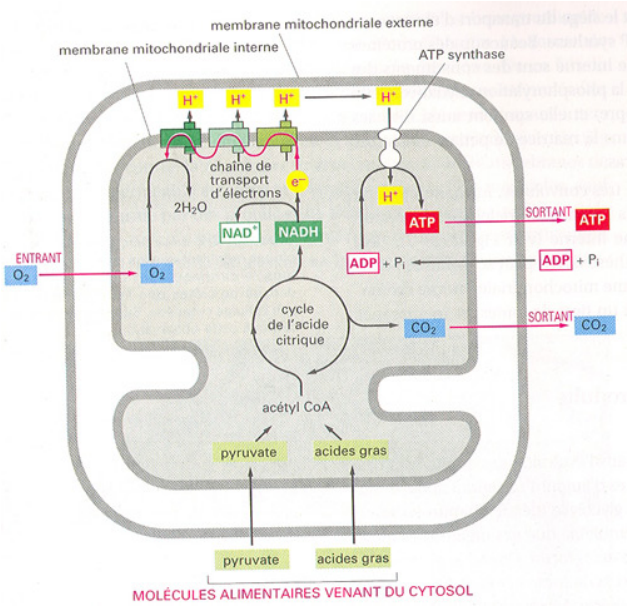
Exemple :
N langues, faut-il N(N-1) traducteurs ou 2(N-1) en choisissant un pivot ?

Structure d’une interface

Plusieurs points de vues

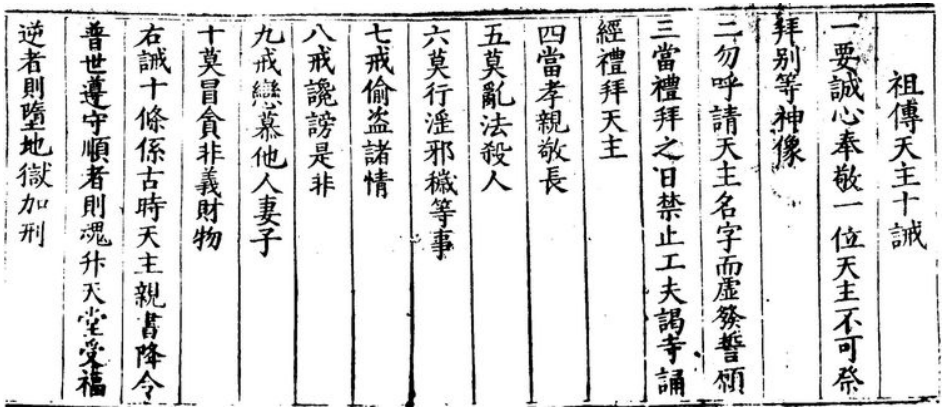
- Physique/chimie
- Science de la communication
- Électronique
- Informatique

Des outils différents de description . . .



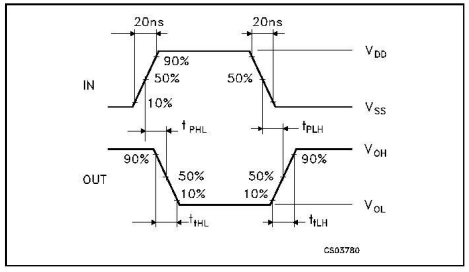
http://www.humans.be/pages/biomitochondrie.htm

1. Échange de composés


祖傳天主十誡
一要誠心奉敬一位天主不可祭拜別等神像
二勿呼請天主名字而虛發誓願
三當禮拜之日禁止工夫謁寺誦經禮拜天主
四當孝親敬長
五莫亂法殺人
六莫行淫邪穢等事
七戒偷盜諸情
八戒讒謗是非
九戒繼慕他人妻子
十莫冒貪非義財物
右誡十條係古時天主親書降令
普世遵守順者則魂升天堂受福
逆者則墮地獄加刑

http://www.matteo-ricci.org/Opera/m1_decalog.html

1. Physique (le support : papier, air, électrons, etc.)
2. Orthographique (le codage : ASCII, unicode, morse, jpeg, etc.)
3. Lexicale (les mots : les séparateurs)
4. Grammaticale (les règles d'ordonnancement)
5. Fonctionnelle (le sens, les usages - droits, etc.)



`http://eicom.ru/pdf/datasheet/ST_Microelectronics_PDFS/HCF40107B/HCF40107B.html`

`http://www.ti.com/lit/ds/symlink/sn74ls00.pdf`

1. Physique (mécanique, broches, etc.)
2. Logique (noms des signaux, sens/direction)
3. Électrique (pilotage tension/courant, niveaux/états, vitesses de commutation, temps de maintien, etc.)
4. Protocolaire

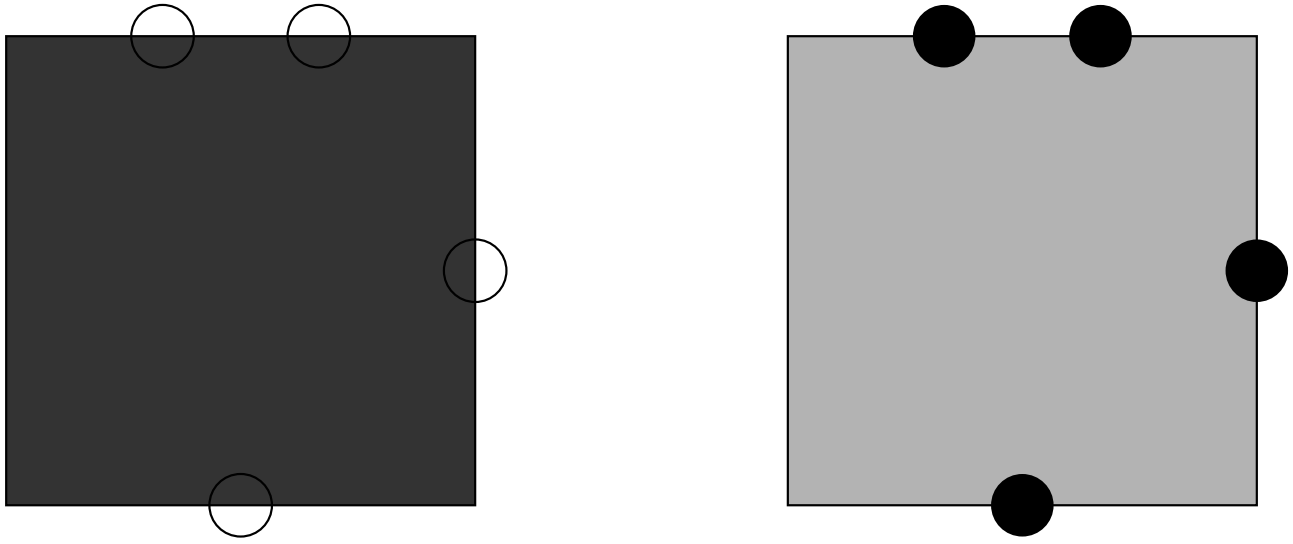
Possible Functions of the Common High-Level API

Category	Function	Description
Management	Open	Open a resource in the system, create a handle to the resource
	Close	Close the resource, free the resource handle
	Reset	Put the resource in a know state
	Execute	Load a resource or cause a process to start
	Claim	Lock access to a shared resource
	Release	Unlock access to a shared resource
Memory Access	MemRead	Read the memory resource (this may be local or remote via a bus)
	MemWrite	Write the memory resource
Link Access	LinkRead	Read the stream resource into a buffer
	LinkWrite	Write to the stream resource from a buffer
Signals	SendSignal	Send a signal to the resource
	WaitSignal	Wait for a signal from the resource
	CallbackConnect	Attach a callback to a resource's signal
	CallbackDisconnect	Detach a callback from a resource's signal

Table 1 Listed here are some of the possible functions of the high-level common layer, which would use something like COM to implement the remappable interfaces to the next layer down, the board services layer. Every device in a system would be given a resource name/handle, allowing it to be addressed and interacted with.

`http://www.rtcmagazine.com/articles/view/100056`

1. Physique (en général abstrait, caché)
2. Logique (noms)
3. Sémantique (sens)
4. Synchronisation (usage, protocole)
5. Qualité de service



Cas simples : un système et son/ses interfaces
Isolé : Encapsulation. Boite noire.
En fait boite grise ; l'interface expose une partie du contenu en choisissant le niveau d'exposition.

Pourquoi ?

- Découpler, isoler, séparer
- Diviser pour régner (Maîtriser la complexité)

Conséquences :

- Simplifier l'usage (utilisateur)
- Comprendre séparément (chercheur)
- Faire développer séparément (ingénieur)

L'identification des bonnes interfaces est liée à l'identification des bonnes frontières.

- Les systèmes physiques présentent souvent des frontières mécaniques naturelles.
- Les systèmes informatiques offrent plus de liberté. (module, class, package, composant, aspect, etc.)

Un guide (en génie logiciel) : forte cohérence - faible couplage ; une responsabilité.

Au-delà des objectifs de ce cours. . . Mais, on fera un peu d'architecture.

Spécification d'une interface

Description de tout ce qui est nécessaire au bon usage du système.

- Mécanique (poids, résistance mécanique, dimensions, nature des matériaux, frottement, etc.)
- Electrique (tension, intensité, sens, évolution du signal, etc.)
- Information (codage, lexique, sens, ordre d'utilisation, qualité de service, etc.)

Les questions : quoi, comment, où, quand, qui doivent trouver une réponse
Et la question *pourquoi* ?

Les mêmes questions ont des réponses de niveaux différents :

- Le client, l'utilisateur
- L'organisateur de la fabrication
- L'architecte
- Le concepteur
- Le fabricant

Organiser les questions (ex : Zachman, DODAF)

	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*
Objective/Scope (contextual) *Role: Planner*	List of things important in the business	List of Business Processes	List of Business Locations	List of important Organizations	List of Events	List of Business Goal & Strategies
Enterprise Model (conceptual) *Role: Owner*	Conceptual Data/ Object Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
System Model (logical) *Role:Designer*	Logical Data Model	System Architecture Model	Distributed Systems Architecture	Human Interface Architecture	Processing Structure	Business Rule Model
Technology Model (physical) *Role:Builder*	Physical Data/Class Model	Technology Design Model	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Detailed Reprentation (out of context) *Role: Programmer*	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Speculation
Functioning Enterprise *Role: User*	Usable Data	Working Function	Usable Network	Functioning Organization	Implemented Schedule	Working Strategy

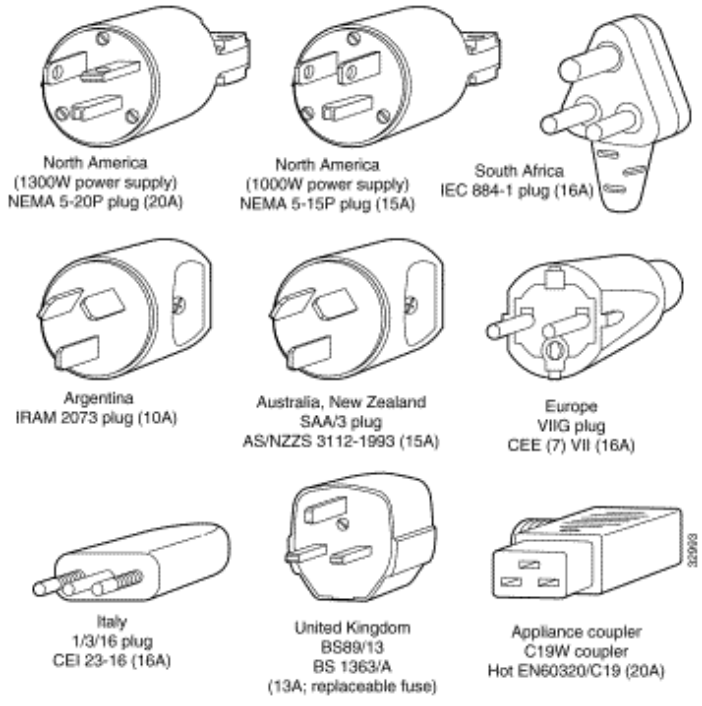
`http://en.wikipedia.org/`

L'*Architecture Framework* de Zachman ne parle pas de :

- coût du service
- responsabilité juridique
- droit d'accès au service
- etc.

- Langue naturelle
- Langage de programmation (API)
- Langages de modélisation (UML, SysML)

But : *décrire pour vérifier la comptabilité des connexions*



Bibliothèque de code C `sqrt(double):double` Qu'est-ce que cela garantit ? La correction de l'édition de lien.

Comment garantir que `sqrt(double):double` calcule bien la racine carré ?

```
sqrt(x:double):double
 pre: x >=0
 post : result >=0 && x = result * result
```

Précondition et postcondition définissent le *sens* de l'opération.

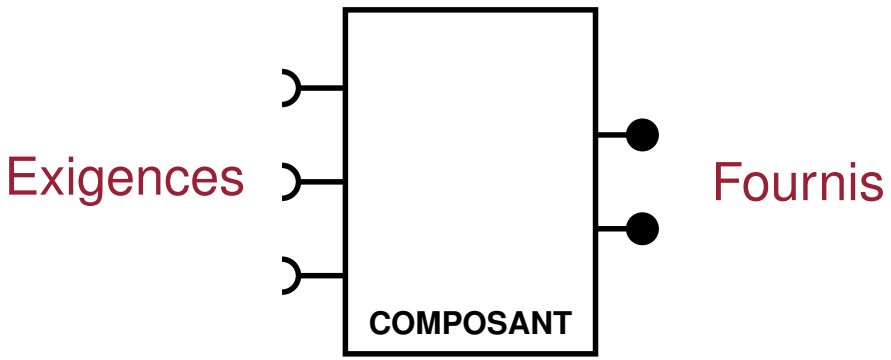
Offre de service

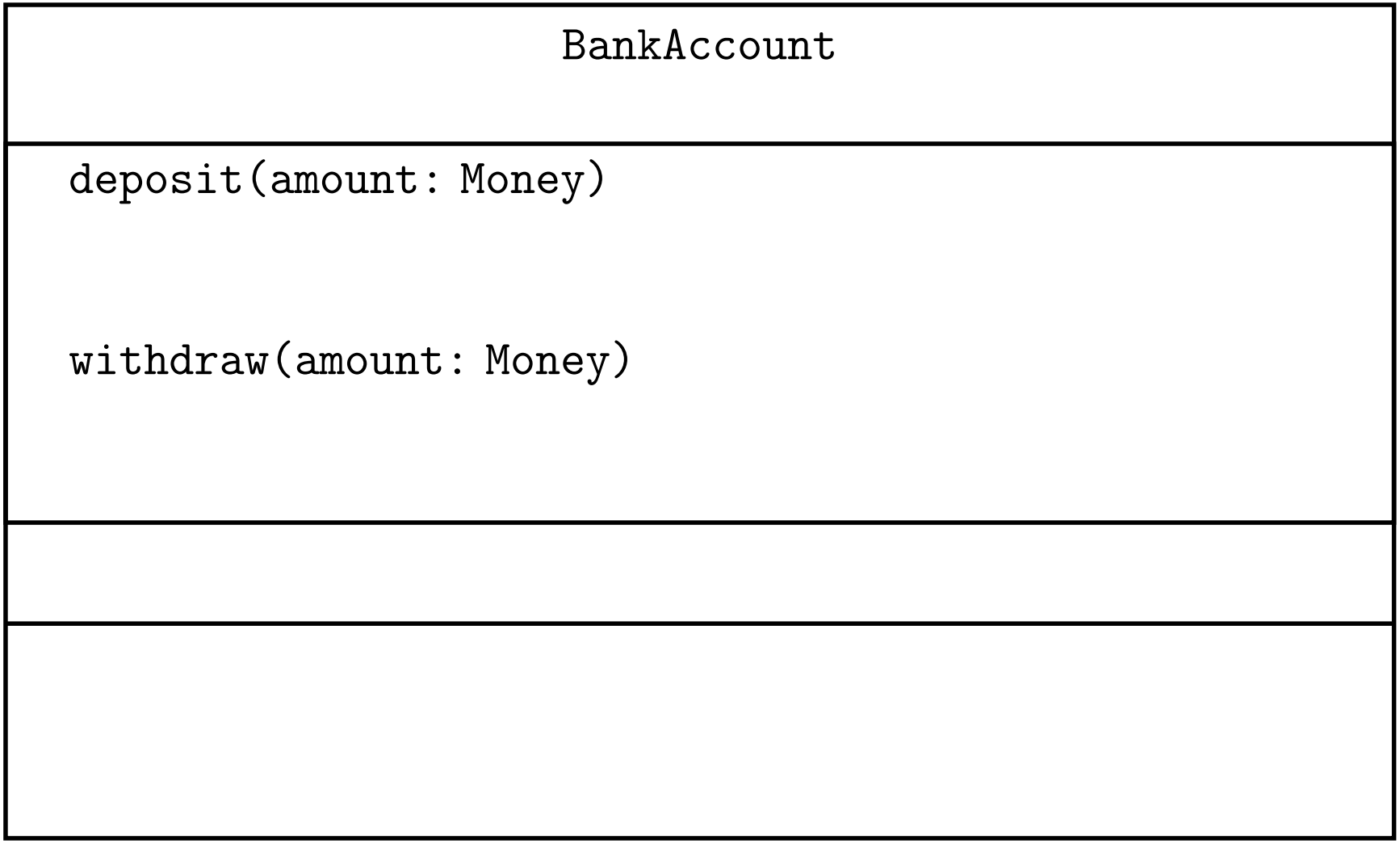
1. Niveau syntaxique : `sqrt(double):double`
2. Niveau sémantique : `pre: post:`
3. Niveau synchronization : règles d'utilisation (concurrence, ordre autorisés, protocole)
4. Niveau QoS : propriétés quantitative (efficacité, fiabilité, disponibilité, etc.)

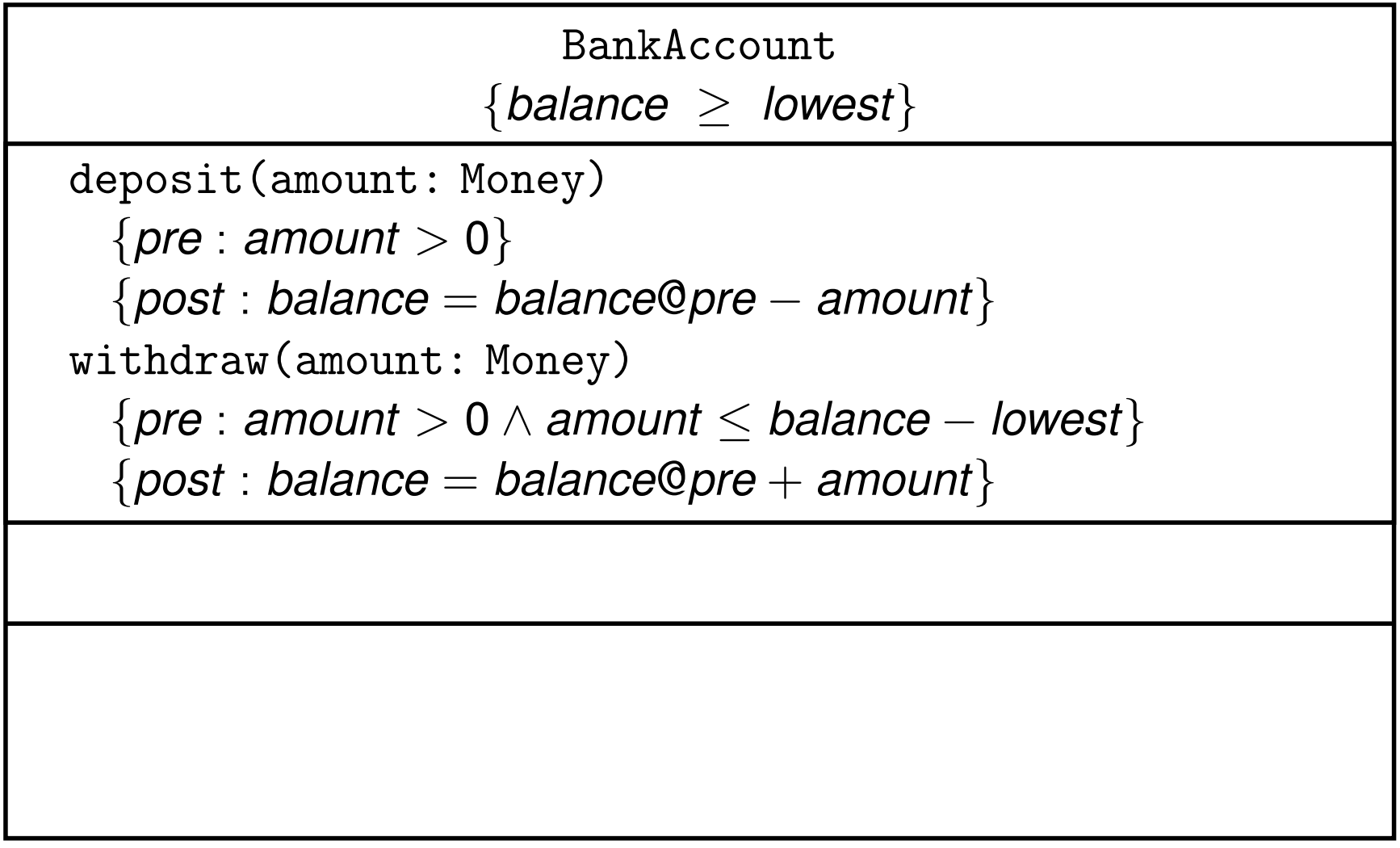
Demande de service

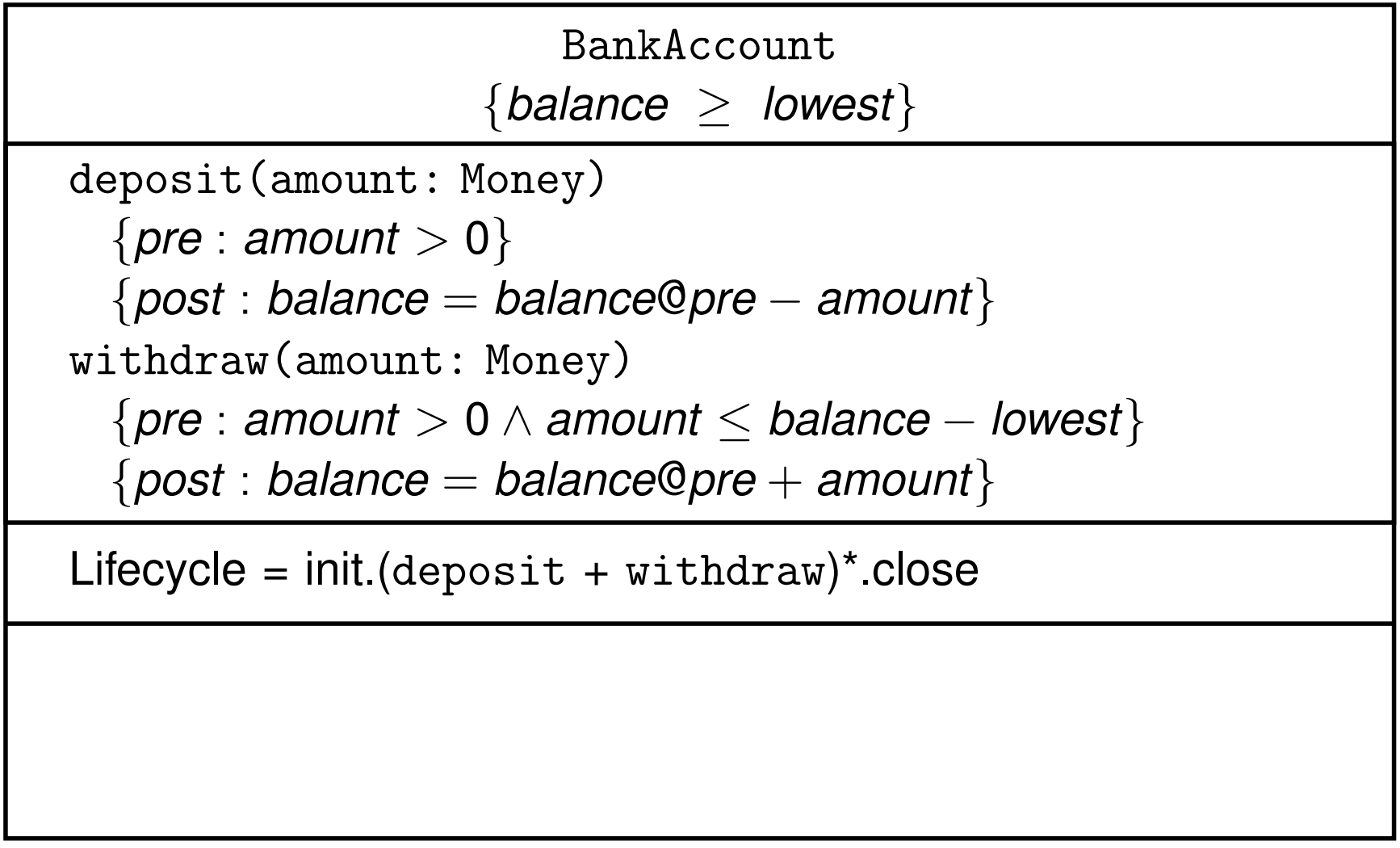
1. Niveau syntaxique : `sqrt(double):double` – même signature
2. Niveau sémantique : `pre: post:` – vérification (statique ou dynamique)
3. Niveau synchronization : règles d'utilisation (concurrence, ordres autorisés, protocole)
4. Niveau QoS : attente du client

- Unité logique (pas forcément de déploiement)
- Qui possède une description explicite :
  - De ce qu'elle fait
  - De ce dont elle a besoin
- \+ ou - formelle (documentation, contrat, . . .)
- Et qui peut être assemblée




BankAccount
deposit(amount: Money)
withdraw(amount: Money)


BankAccount
{balance ≥ lowest}
deposit(amount: Money)
{pre : amount > 0}
{post : balance = balance@pre − amount}
withdraw(amount: Money)
{pre : amount > 0 ∧ amount ≤ balance − lowest}
{post : balance = balance@pre + amount}


BankAccount
{balance ≥ lowest}
deposit(amount: Money)
{pre : amount > 0}
{post : balance = balance@pre − amount}
withdraw(amount: Money)
{pre : amount > 0 ∧ amount ≤ balance − lowest}
{post : balance = balance@pre + amount}
Lifecycle = init.(deposit + withdraw)*.close

`BankAccount` $\{balance \geq lowest\}$

`deposit(amount: Money)`
$\{pre : amount > 0\}$
$\{post : balance = balance@pre - amount\}$
`withdraw(amount: Money)`
$\{pre : amount > 0 \wedge amount \leq balance - lowest\}$
$\{post : balance = balance@pre + amount\}$

Lifecycle = init.(`deposit + withdraw`)*.close
ResponseTime(`deposit`) $< 1s$ when *Users* < 1000 *ResponseTime*(`withdraw`) $< 1s$ when *Users* < 1000 Availability(`BankAccount`) all days from 1:00 to 0:00

- Contrat syntaxique : langages - types (IDL, . . .)
- Contrat sémantique : OCL, assertions, Eiffel, JML
- Contrat de synchronisation : Automates, Protocol State Machine, logiques temporelles, . . .
- Contrat de qualité de service : QML

Problèmes

- Spécifier avec précision
  - . . . vient d'être vu
  - Quels outils ?
- Vérifier les assemblages
  - statiquement (essentiellement les systèmes fermés)
  - dynamiquement (essentiellement les systèmes ouverts)
  - Quels outils ?
- Assurer l'évolution
  - un changement d'interface impacte les 2 sous-systèmes reliés
  - découpler spécification - implémentation
  - Quels outils ?

Conclusion sur les interfaces

L'interface est un contrat : statique et dynamique.

- Essentiel pour spécifier des systèmes – intermédiation
- Facilite l'utilisation – mode d'emploi
- Facilite l'assemblage
- Contraint/guide la conception

La notion d'interface est au cœur de l'ingénierie système.
La notion d'interface est au cœur de l'étude des interactions.

Exemple historique

On identifie :

- des fonctions, des modules
- des interactions
- des données (table, arbre)
- des entrées, des sorties

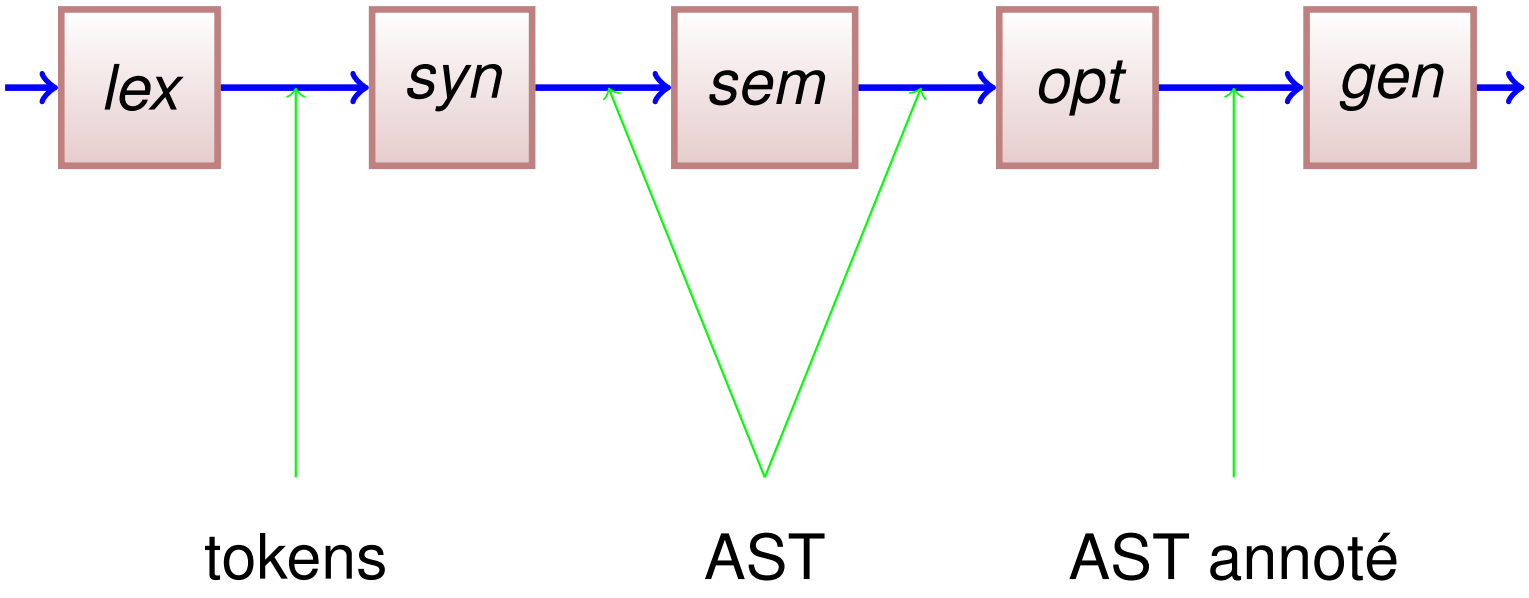
Comment abstraire ?

- des boîtes
- des traits

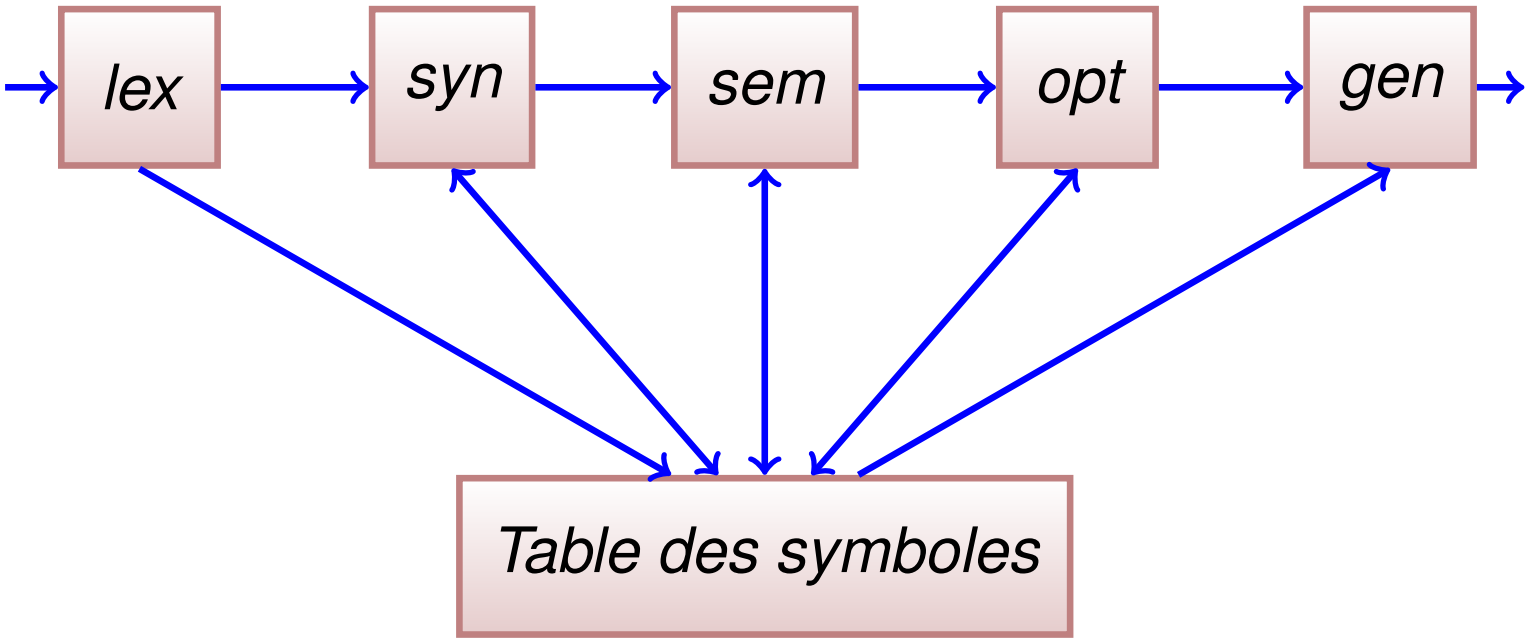
. . . et leurs propriétés . . .

Exemple issu de « An Introduction to Software Architecture », Garlan et Shaw, 1993 [GS93]

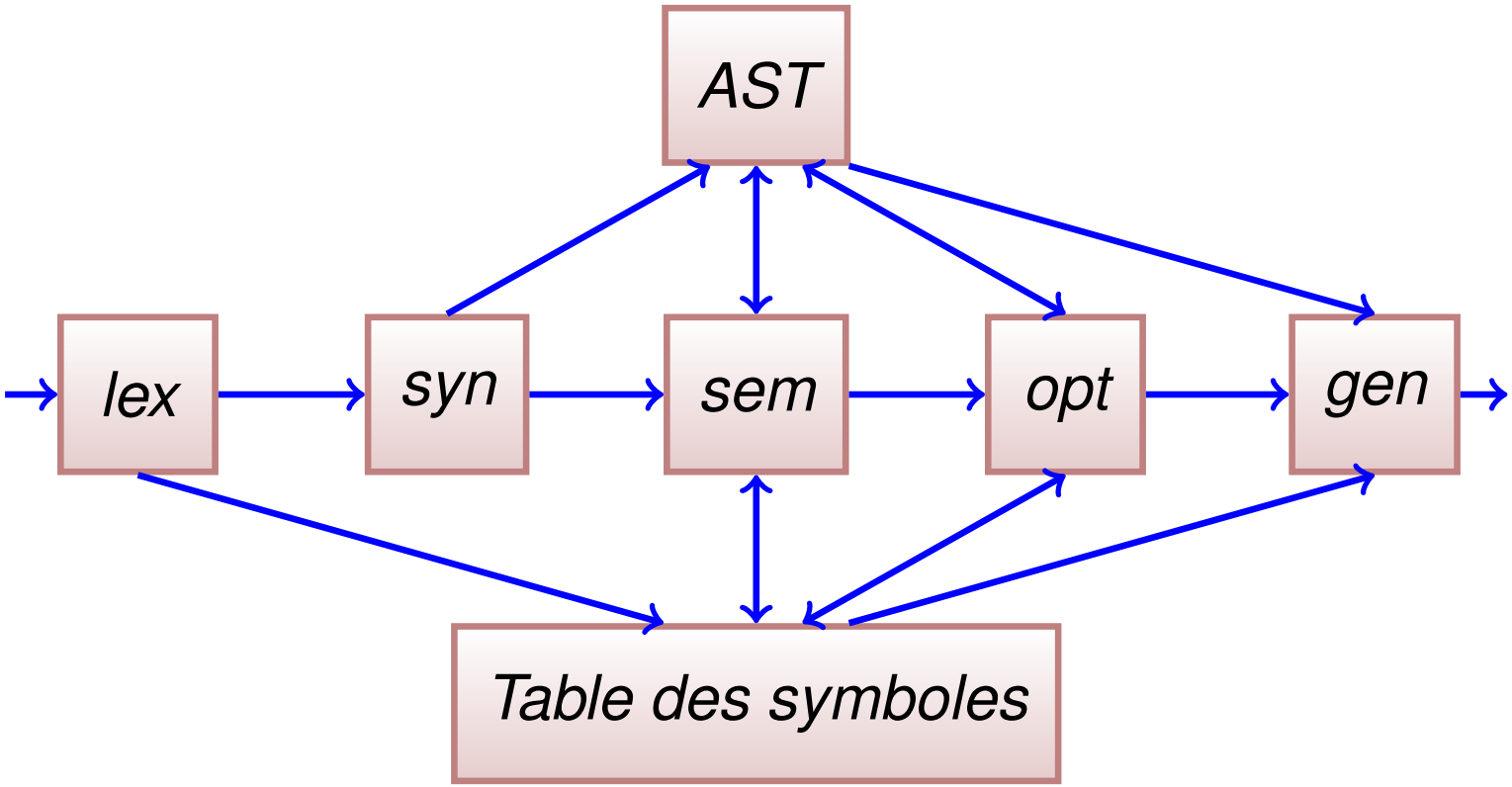
Une séquence de fonctions. . .



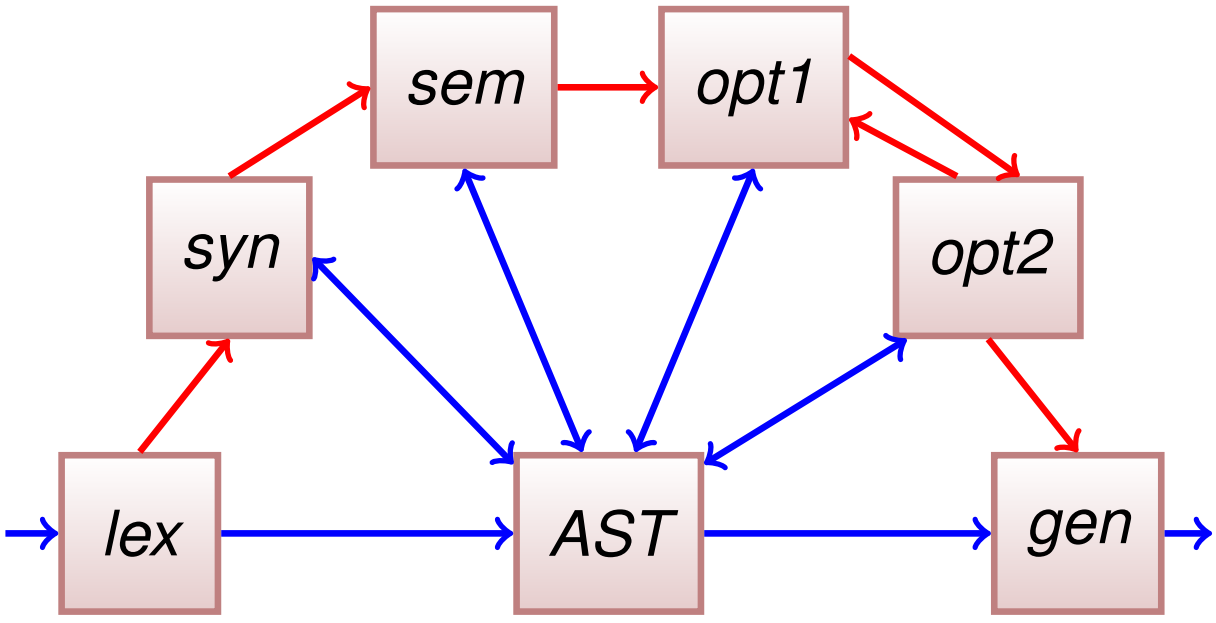
Une séquence de fonctions qui partagent une table de symboles. . .



Une séquence de fonctions qui partagent une table de symboles et un arbre syntaxique abstrait (AST)...



Une séquence de fonctions qui partagent des informations et enchaine des traitements optionnels. . .



On peut déjà dire beaucoup de choses avec la topologie : partage, goulot d'étranglement, point de défaillance, etc.

Plusieurs styles

- batch
- tableau noir

Des propriétés différentes : efficacité, évolutivité

Architecture

Une architecture est composée de :

- composants (les boîtes)
- connecteurs (les traits)

pour former une configuration.

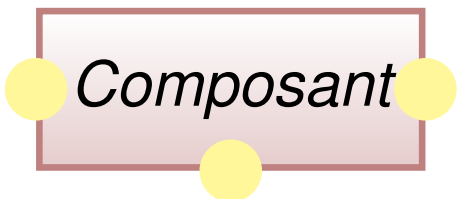
> Architecture = structure avec des éléments et des relations

Pas de définition standard

- `http://www.sei.cmu.edu/architecture/definitions.html` ou
- `http://en.wikipedia.org/wiki/Software_architecture`

Pas de définition standard (mais de nombreuses définitions que nous verrons plus tard.)

- une unité (la boîte) délimitée par
- une frontière (la bordure) qui possède
- des points d'accès (ports)



Composant ?

Que représente un composant ?
Diverses “unités” . . .

- une fonction, une procédure, un calcul
- un objet
- un service
- une unité de stockage
- une interface
- . . .

Un bon guide est d’y associer une responsabilité

Encore moins de définition standard (mais de nombreuses définitions que nous verrons plus tard.)

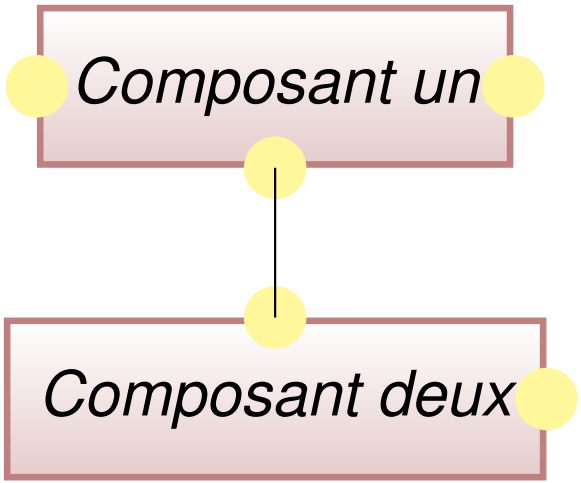
- une connexion qui relie
- deux (ou plusieurs) composants qui
- jouent un role (dans la connexion)

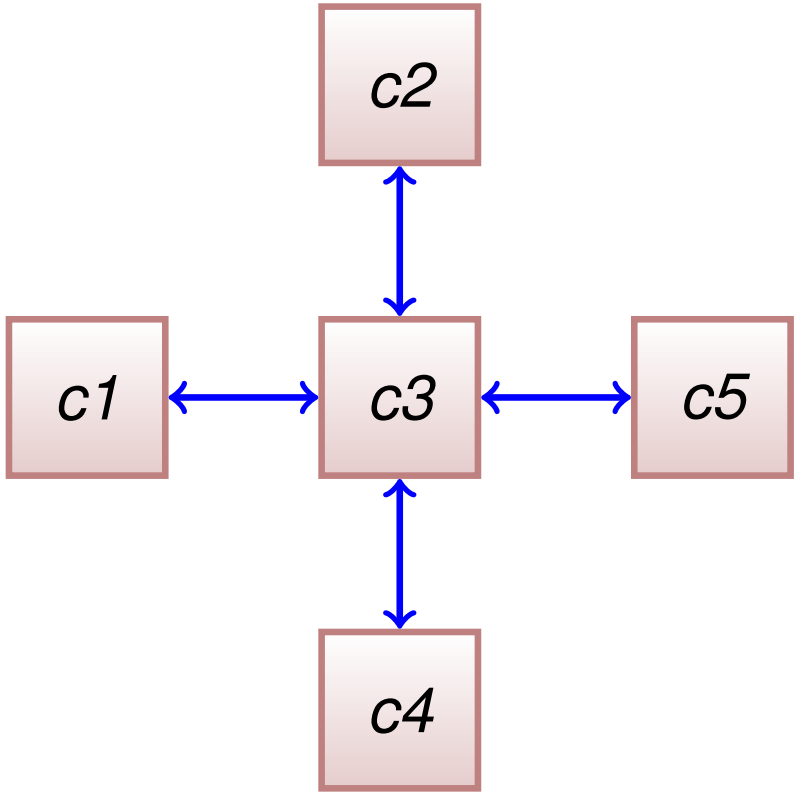
Que représente un connecteur ?
Divers “moyens de connection” . . .

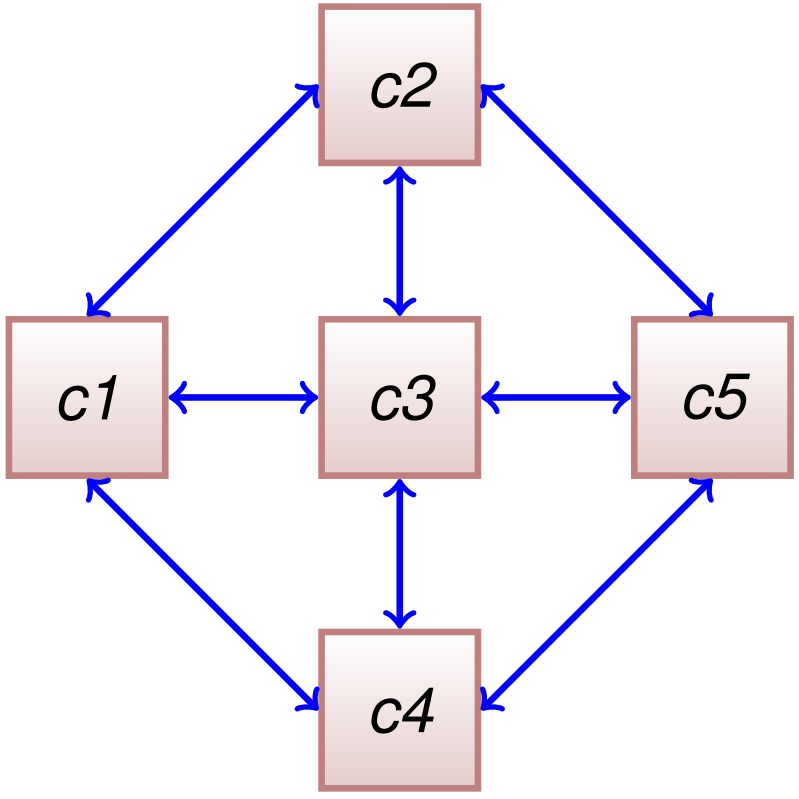
- un bus
- un protocol
- un appel de procédure (à distance)
- . . .

Un bon guide est d’y associer une transfert de donnée ou de contrôle

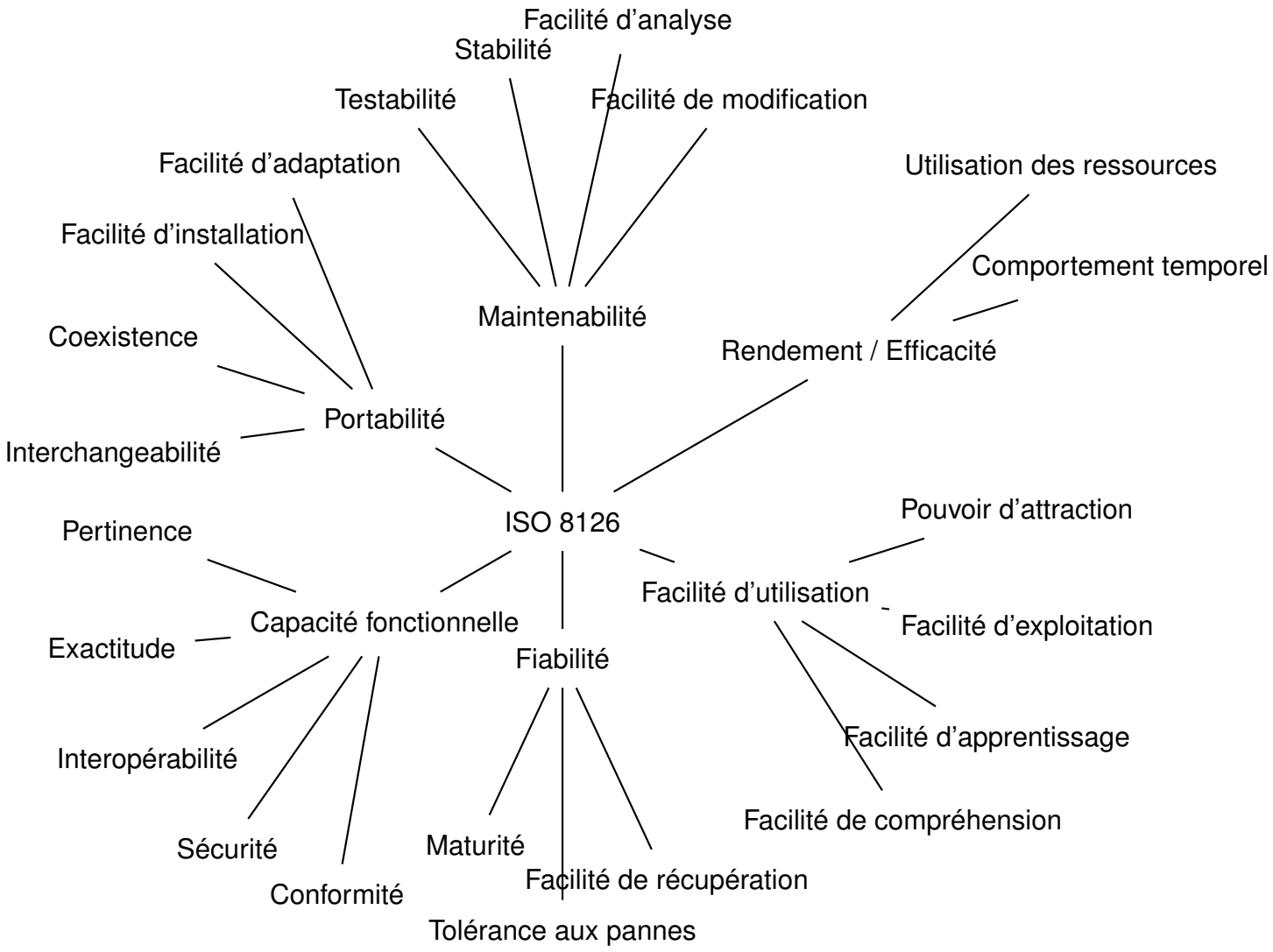
Un ensemble de composants et de connecteurs assemblés.
Les ports sont associés à des rôles. (Quid de la compatibilité ?)




c2
c1
c3
c5
c4

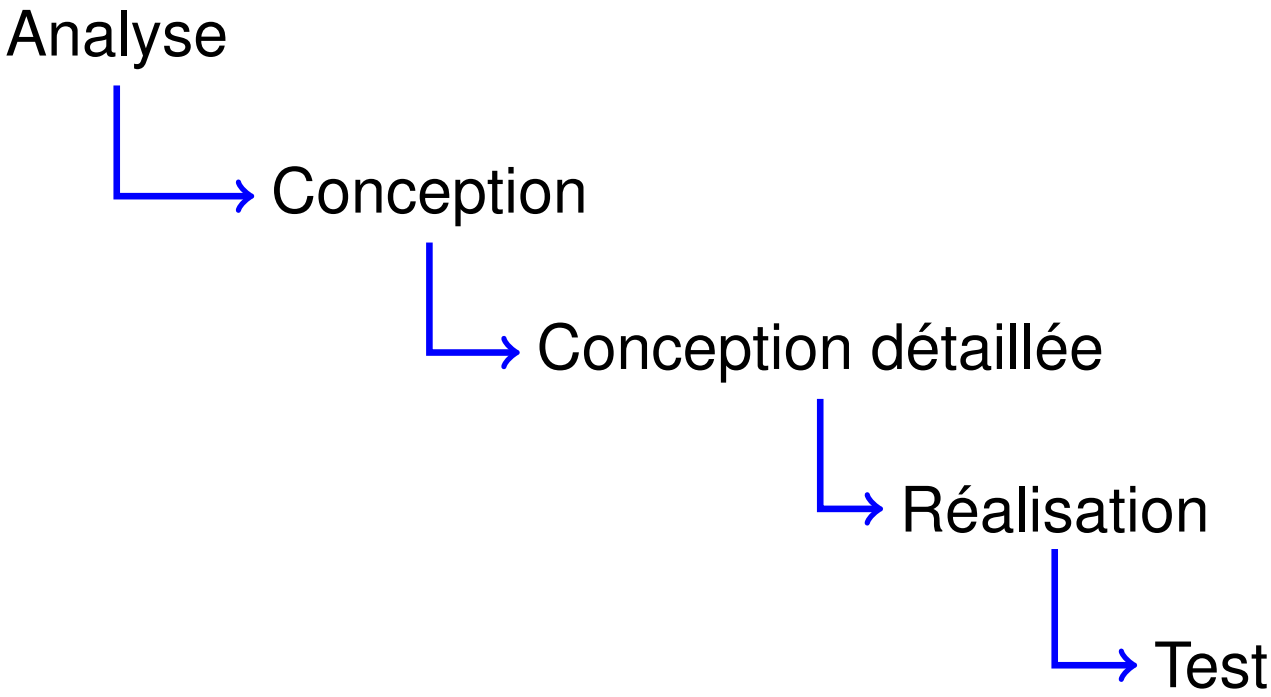

c2
c1
c3
c5
c4

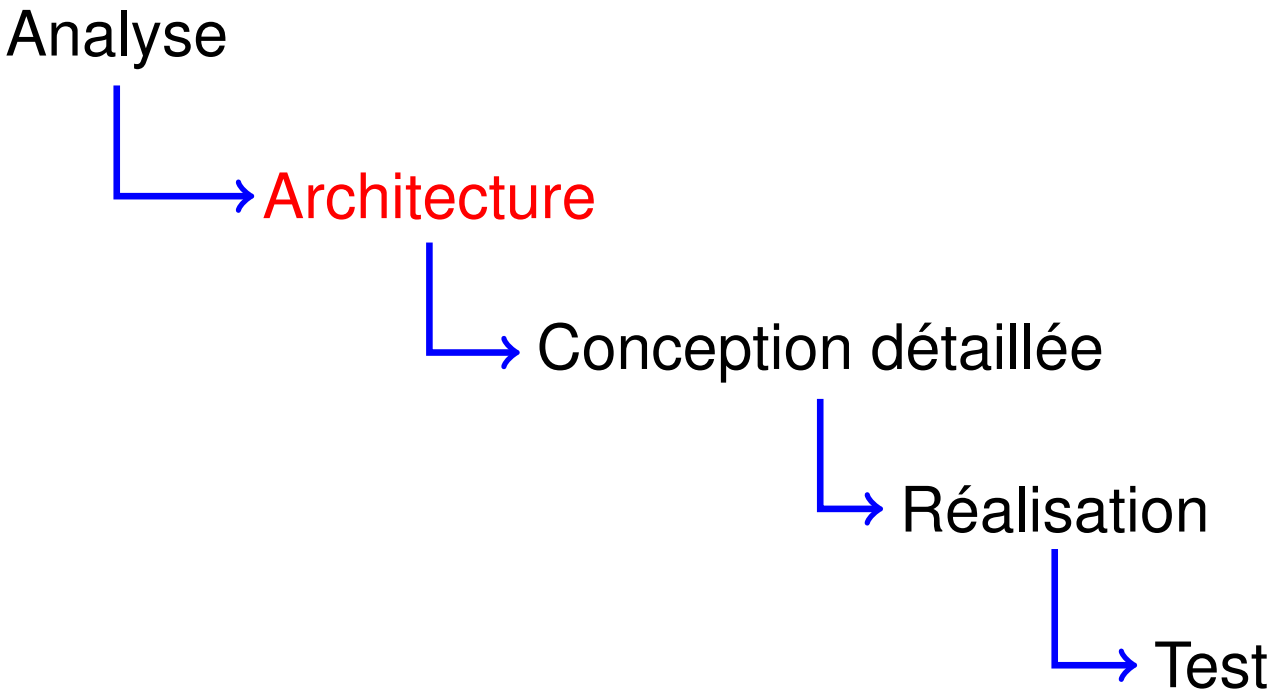
- Couplage (objectif faible)
- Cohérence (objectif forte)
- Robustesse
- ...


ISO 8126
Maintenabilité
Facilité d'analyse
Stabilité
Testabilité
Facilité de modification
Portabilité
Facilité d'adaptation
Facilité d'installation
Coexistence
Interchangeabilité
Rendement / Efficacité
Utilisation des ressources
Comportement temporel
Capacité fonctionnelle
Pertinence
Exactitude
Interopérabilité
Sécurité
Conformité
Fiabilité
Maturité
Facilité de récupération
Tolérance aux pannes
Facilité d'utilisation
Pouvoir d'attraction
Facilité d'exploitation
Facilité d'apprentissage
Facilité de compréhension

- ACME
- Aesop
- Darwin
- MetaH
- Rapide
- Wrigth
- AADL

Très bon survol dans [MT00].


Analyse
Conception
Conception détaillée
Réalisation
Test


Analyse
Architecture
Conception détaillée
Réalisation
Test

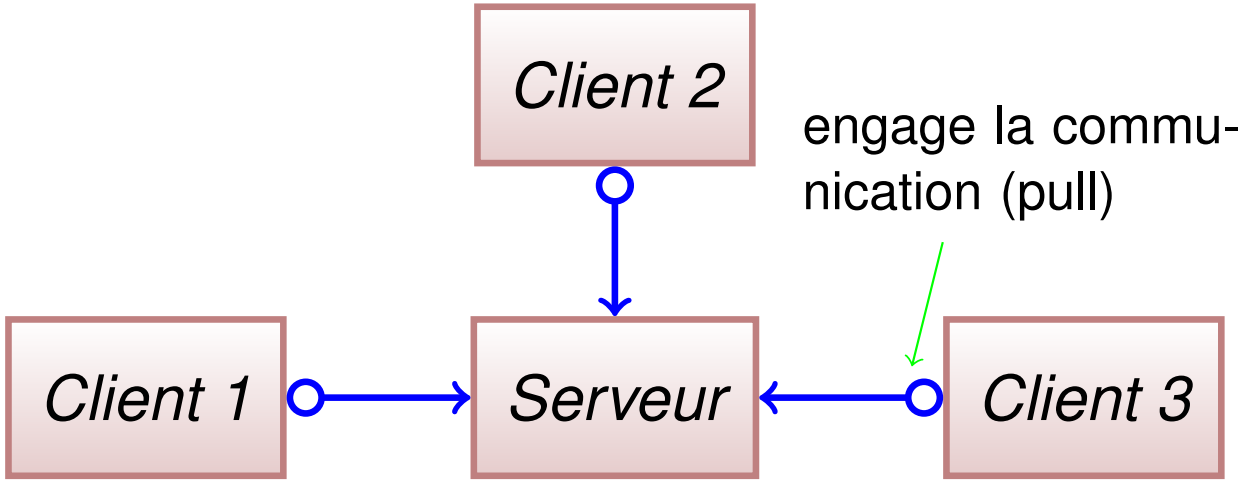
Styles d’architecture

Ensemble de types de composants et de types de connecteurs.

Règles d’assemblages à respecter.

Ensemble de types de composants et de types de connecteurs.
ex. Client, Serveur

Règles d'assemblages à respecter.



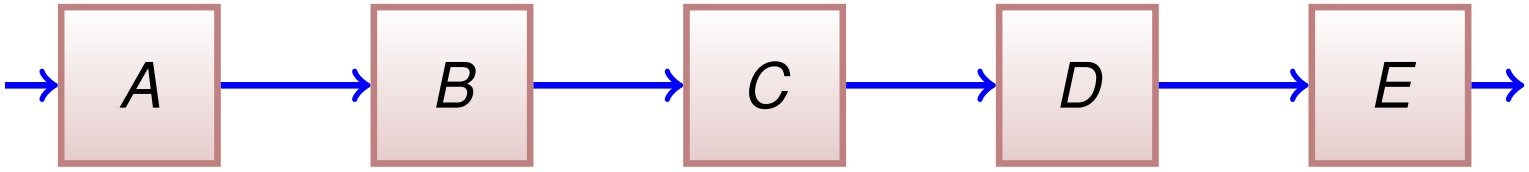
- Dataflow
- Pipe & Filter
- Tableau noir
- Publish-subscribe
- En couches
- Client-Serveur
- Pair-à-pair
- N-tiers

- Capitaliser du savoir-faire, des connaissances
- Définir du vocabulaire
- Améliorer les capacités de communication (entre personnes)
- Associer des propriétés (avantage/inconvénient) aux styles

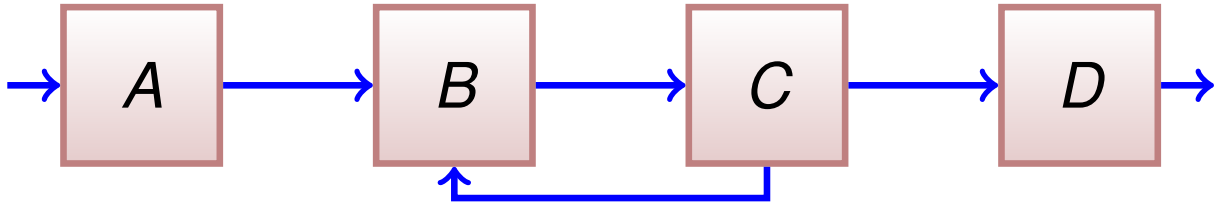
Connaissez-vous d'autres situations comparable ?

- Capitaliser du savoir-faire, des connaissances
- Définir du vocabulaire
- Améliorer les capacités de communication (entre personnes)
- Associer des propriétés (avantage/inconvénient) aux styles

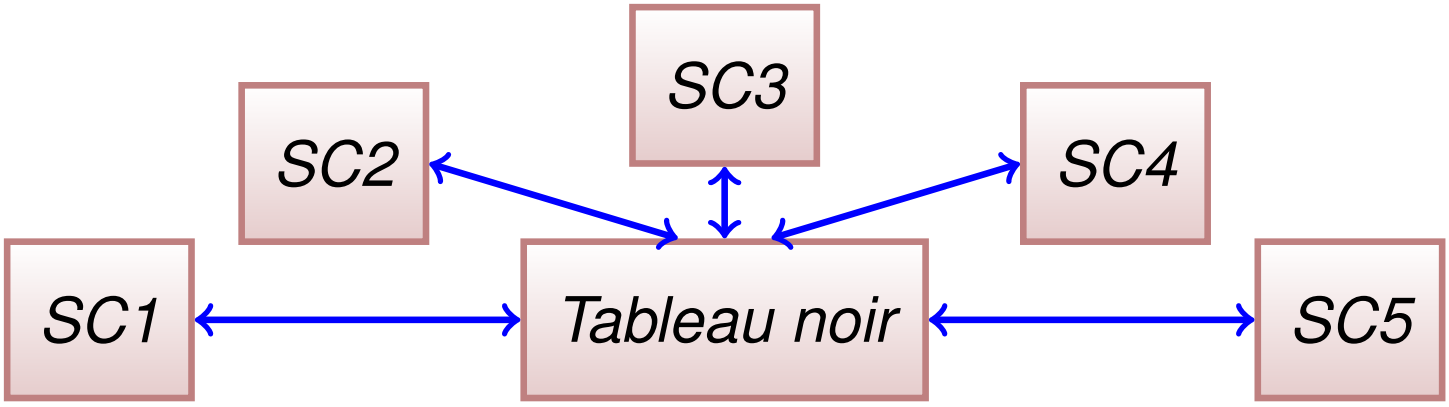
Connaissez-vous d'autres situations comparable ?
Les patrons (de conception, d'organisation, . . .)



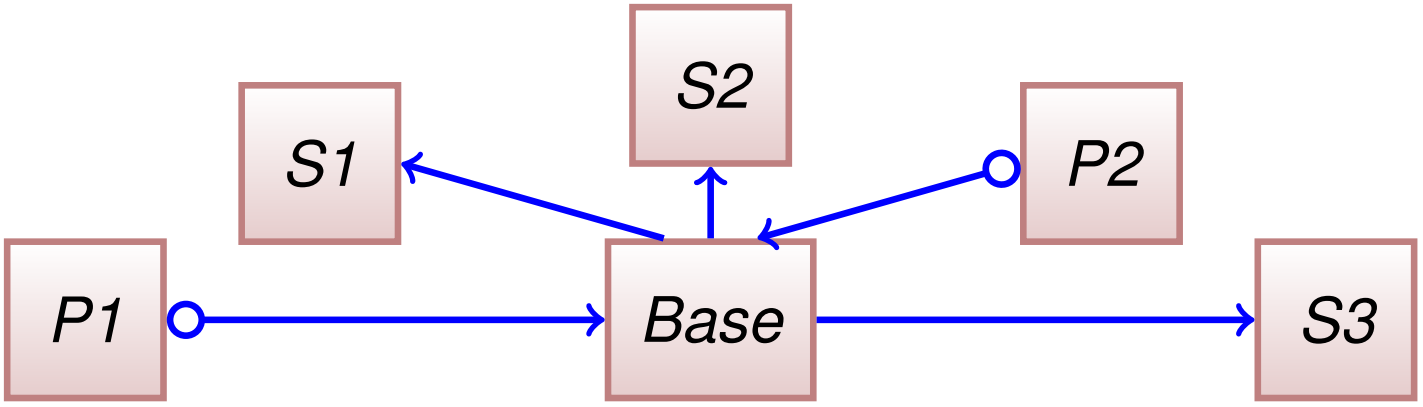
Composants	*Connecteurs*
Filtre	Transfert de donnée mono-directionnel
Avantages	*Inconvénients*
Composition simple Couplage faible Boites interchangeables	1 entrée, 1 sortie seulement Plus simple avec des données homogènes



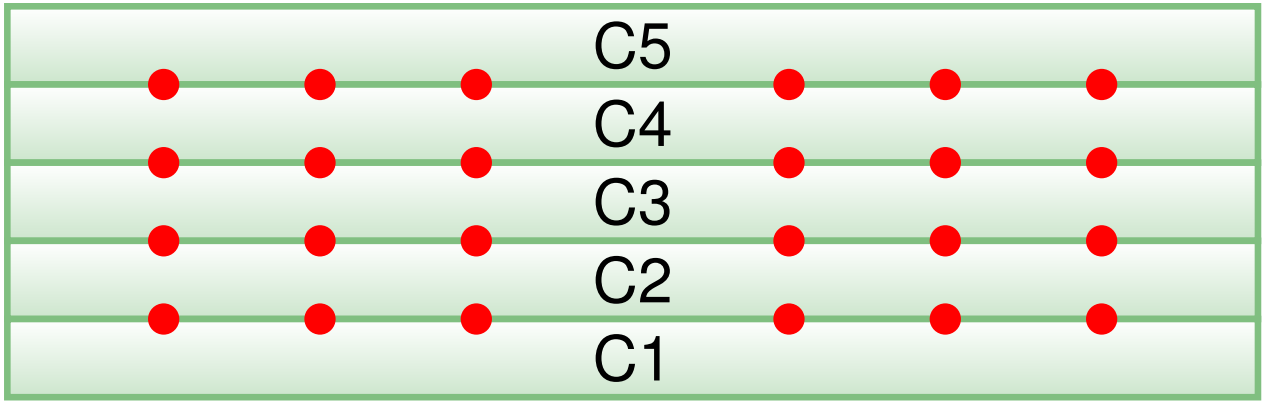
Composants	*Connecteurs*
Filtre	Pipe (avec buffer)
Avantages	*Inconvénients*
Composition simple Couplage faible Boites interchangeables	Risques d'interblocage Plus simple avec des données homogènes Buffer de taille finie



Composants	*Connecteurs*
Tableau noir Source de connaissance	protocole bi-directionnel
Avantages	*Inconvénients*
Correction des erreurs rapide Peu de perte information Extensible Couplage faible Composition simple	Risque d'accumulation de données inutiles Contraintes de vocabulaire Dépendant du domaine d'application

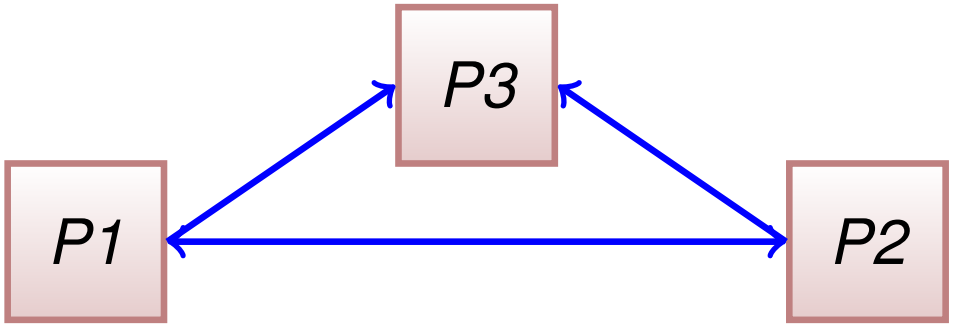


Composants	*Connecteurs*
Base Publisher Subscriber	protocole orienté message
Avantages	*Inconvénients*
Anonymat Couplage faible Extensible	Performance Asynchrone (?)



Composants	*Connecteurs*
Couche	Appel de procédure
Avantages	*Inconvénients*
Conception incrémentale Maintenance Réutilisation	Taille des niveaux Efficacité

Composants	*Connecteurs*
Serveur Client	protocole *pull*
Avantages	*Inconvénients*
Composition simple Couplage faible Extensible Efficace	Point à point seulement



Composants	*Connecteurs*
Pair	protocole dédié
Avantages	*Inconvénients*
Symmétrie Résistance aux pannes Extensible	Performance

Une variante du “client-serveur” ou de “en couches”.
Un tiers pour une responsabilité.
Exemple :

1. Tiers de persistance
2. Tiers métier
3. Tiers de présentation

- Les styles N-tiers, en couche se ressemblent ; la nature de la connexion est différente : plus souple en N-tiers.
- Les styles tableau noir, publish subscribe se ressemblent ; la nature des rôles des composants qui utilisent la zone partagée est plus ou moins spécialisée.
- Les styles 2-tiers, client serveur, tableau noir, ou publish-subscribe se ressemblent . . .

Un système complexe utilise plusieurs styles :

Horizontalement À un niveau donné, une partie du système peut être d'un style et une autres partie d'un autre style

Verticalement Quand on étudie la structure interne d'un composant d'un style, on peut y trouver un autre style.

Interaction et architecture

Une architecture définit une topologie d'interaction.

Elle montre (ou peut montrer) :

- la topologie
- les couplages
- les lieux des interfaces
- le sens des échanges
- les initiatives des échanges

Elle cache :

- le détail des interfaces
- l'ordre des échanges (aspect temporel)

References I

P Eugster, P Felber, R Guerraoui, and A Kermarrec.

The Many Faces of Publish/Subscribe.

ACM Computing Surveys (CSUR), 2003.

David Garlan and M Shaw.

An Introduction to Software Architecture.

In *Advances in Software Engineering and Knowledge Engineering*, pages 1–40. 1993.

Nenad Medvidovic and Richard N Taylor.

A classification and comparison framework for software architecture description languages.

IEEE Transactions on Software Engineering, 2000.