

Assistance à l'architecte pour la construction d'architectures à base de composants

Nicolas Desnos¹ Christelle Urtado¹ Sylvain Vauttier¹
Marianne Huchard²

¹LGI2P - Ecole des Mines d'Alès
Parc Scientifique G. Besse - F30035 Nîmes

²LIRMM - UMR 5506
CNRS et Univ. Montpellier 2 - F34392 Montpellier cedex 05

Plan

- 1 Aide à la construction d'architectures valides satisfaisant un objectif fonctionnel
- 2 Validité d'une architecture
- 3 Construction d'architectures potentiellement valides satisfaisant un objectif fonctionnel
 - Quelles interfaces connecter ?
 - Où connecter les interfaces ?
 - Quand arrêter les connexions ?
- 4 Conclusion et perspectives

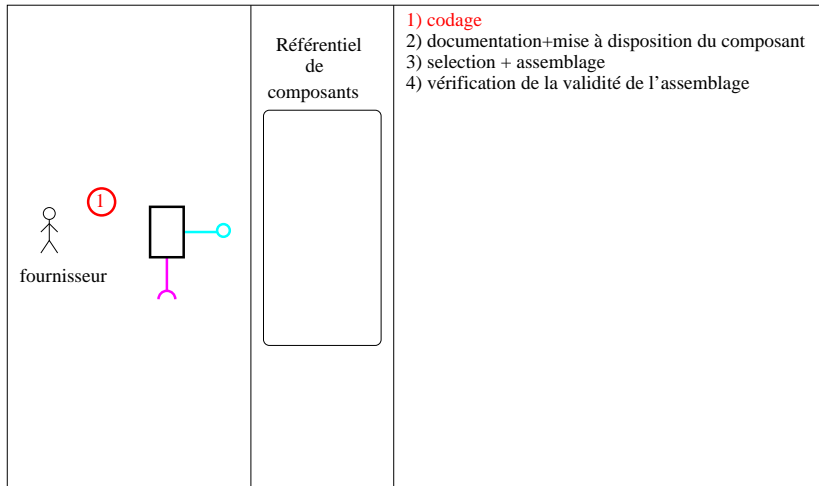
Contexte et définition

- Génie logiciel
 - Développement à base de composants
- Composant = boîte noire + interfaces requises et fournies
- Assemblage de composants = connexions d'interfaces

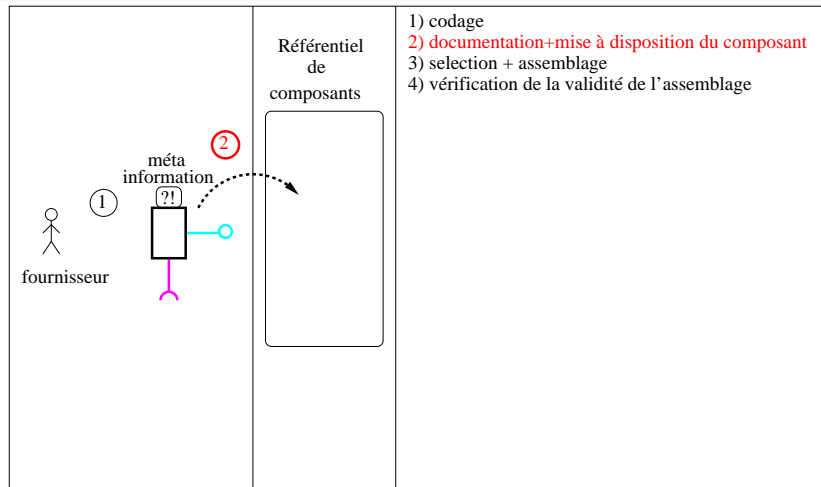
Processus de développement d'une application à base de composants

- 2 Acteurs :
 - Fournisseur : développe + documente des composants
 - Architecte : sélectionne + assemble des composants

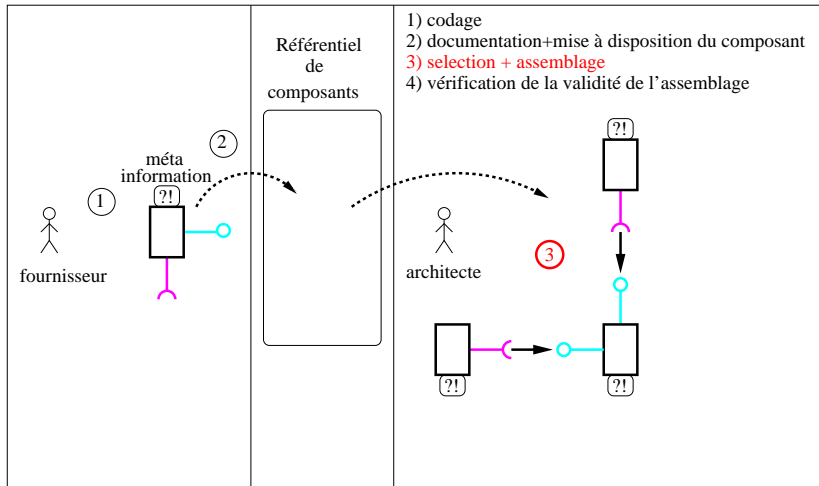
Processus de développement d'une application à base de composants



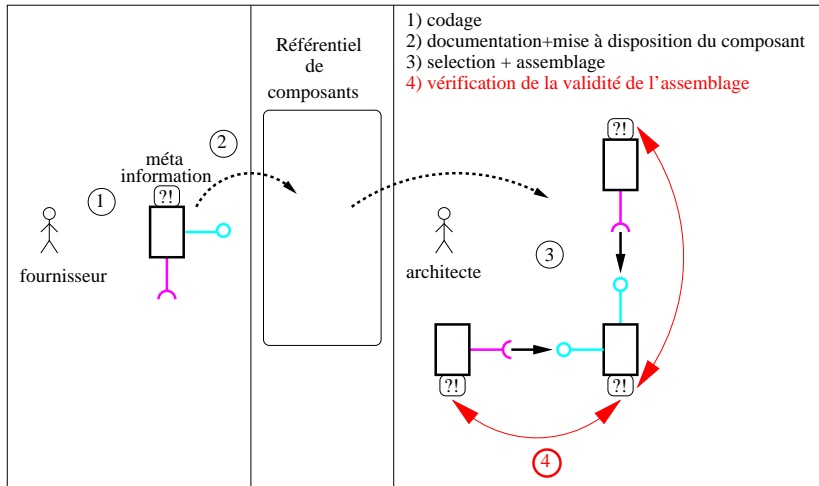
Processus de développement d'une application à base de composants



Processus de développement d'une application à base de composants



Processus de développement d'une application à base de composants



Problématique

- Construction d'architectures valides satisfaisant un objectif fonctionnel
 - Validité = syntaxique + sémantique (protocoles)
 - Objectif fonctionnel = fonctionnalités que l'on souhaite utiliser

Plan

- 1 Aide à la construction d'architectures valides satisfaisant un objectif fonctionnel
- 2 Validité d'une architecture**
- 3 Construction d'architectures potentiellement valides satisfaisant un objectif fonctionnel
 - Quelles interfaces connecter ?
 - Où connecter les interfaces ?
 - Quand arrêter les connexions ?
- 4 Conclusion et perspectives

Validité d'une architecture

- Niveau syntaxique
 - compatibilité des types des interfaces
 - interface requise super-type interface fournie
 - [OMG a, BRU 04, OPL 03]
- Niveau sémantique
 - Compatibilité des protocoles
 - protocole = expression régulière / machines à états
 - compatibilité = comparaisons de machines à états
- Travaux = vérification sémantique d'un assemblage existant
 - [Plà 02, HAC 04 , FAR 02, PRO 03, CAR 03, BOE 02]

Bilan sur la validité d'une architecture

- Vérifier la compatibilité des protocoles → tâche coûteuse [INV 0,ALF 01]
- Impossible à utiliser systématiquement dans un processus de construction d'architectures
- Notre démarche :
 - 1 Construction (sélection) d'architectures potentiellement valides satisfaisant un objectif fonctionnel
 - 2 Application des algorithmes de vérification de la validité sémantique

Plan

- 1 Aide à la construction d'architectures valides satisfaisant un objectif fonctionnel
- 2 Validité d'une architecture
- 3 Construction d'architectures potentiellement valides satisfaisant un objectif fonctionnel**
 - Quelles interfaces connecter ?
 - Où connecter les interfaces ?
 - Quand arrêter les connexions ?
- 4 Conclusion et perspectives

Vers la construction d'architectures valides satisfaisant un objectif fonctionnel

- Point de départ :
 - Objectif fonctionnel (ensemble d'interfaces)
 - Un composant (contient l'objectif fonctionnel)
- Résultat : des architectures
 - Permettant chacune l'utilisation des fonctionnalités de l'objectif fonctionnel
 - Valides

Algorithme naïf de construction d'architectures valides satisfaisant un objectif fonctionnel

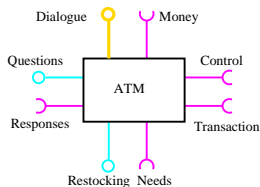
- Construire *toutes les architectures candidates* (sur la seule information des types) et tester leur validité : explosion combinatoire

Réduire la combinatoire

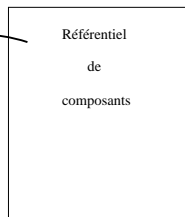
- Exigence : éviter l'explosion combinatoire du processus de construction
- Questions pour atteindre ce but :
 - Quelles interfaces connecter ?
 - Où connecter les interfaces ?
 - Quand arrêter les connexions ?

Exemple

Point de départ



choix
d'un composant



Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Plan

- 1 Aide à la construction d'architectures valides satisfaisant un objectif fonctionnel
- 2 Validité d'une architecture
- 3 Construction d'architectures potentiellement valides satisfaisant un objectif fonctionnel**
 - Quelles interfaces connecter ?
 - Où connecter les interfaces ?
 - Quand arrêter les connexions ?
- 4 Conclusion et perspectives

1er niveau d'information : les interfaces

Quelles informations peut-on utiliser ?

- Les types : information sur les possibilités de connexions

Comment procéder ?

- Tester toutes les possibilités de connexions et vérifier la validité à chaque fois
 - Combinatoire (vérifier au final la validité de toutes les architectures candidates à la validité)

2ème niveau d'information : exploitation des protocoles

Quelles informations peut-on utiliser ?

- Protocole de comportement : exprime toutes les utilisations valides d'un composant
- Scénario d'utilisation : exprime une utilisation particulière d'un composant

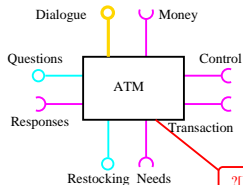
Comment procéder ?

- Déterminer les dépendances entre interfaces (celles qui se trouvent dans un scénario d'utilisation contenant l'objectif fonctionnel) et extraire ces interfaces

Inconvénient

- Processus lourd (information non explicite)

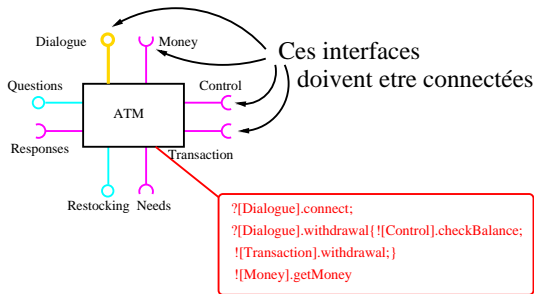
2ème niveau d'information : exploitation des protocoles



```
?[Dialogue].connect;  
?[Dialogue].withdrawal{ ![Control].checkBalance;  
! [Transaction].withdrawal; }  
! [Money].getMoney
```

Money	getMoney getTicket
	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

2ème niveau d'information : exploitation des protocoles



Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

2ème niveau d'information : exploitation des protocoles

Quelles informations peut-on utiliser ?

- Protocole de comportement : exprime toutes les utilisations valides d'un composant
- Scénario d'utilisation : exprime une utilisation particulière d'un composant

Comment procéder ?

- Déterminer les dépendances entre interfaces (celles qui se trouvent dans un scénario d'utilisation contenant l'objectif fonctionnel) et extraire ces interfaces

Inconvénient

- Processus lourd (information non explicite)

3ème niveau d'information : Port

Caractéristiques

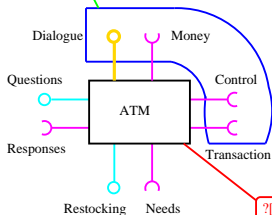
- Port : méta-information
- Port = ensemble d'interfaces
 - [OMG b, ALD 02, HAC 04, BOE 02, PRO 03]
- Regroupe les interfaces qui participent à une collaboration avec un autre composant
- Conforme à un scénario d'utilisation
- Facilite la connexion entre composants

Analyse

- Réduction importante de la combinatoire (connexions entre ports)
- Contrainte de connexion trop importante (peu de solutions)

3ème niveau d'information : Port

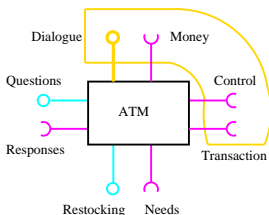
On regroupe les interfaces dépendantes dans un Port



```
?[Dialogue].connect;  
?[Dialogue].withdrawal{![Control].checkBalance;  
![Transaction].withdrawal;}  
![Money].getMoney
```

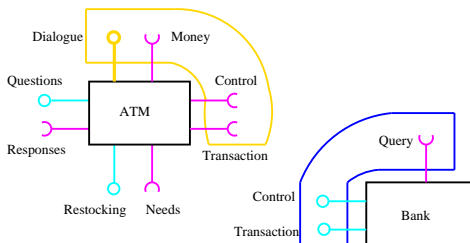
Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

3ème niveau d'information : Port



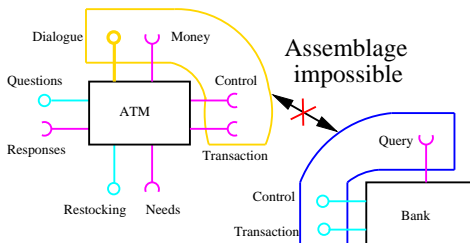
Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

3ème niveau d'information : Port



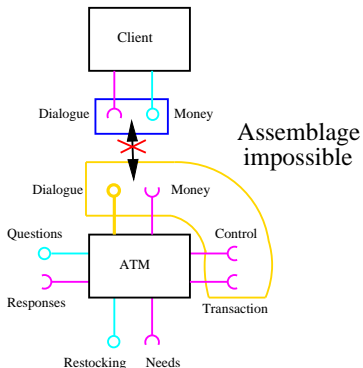
Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

3ème niveau d'information : Port



Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

3ème niveau d'information : Port



Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

3ème niveau d'information : Port

Caractéristiques

- Port : méta-information
- Port = ensemble d'interfaces
 - [OMG b, ALD 02, HAC 04, BOE 02, PRO 03]
- Regroupe les interfaces qui participent à une collaboration avec un autre composant
- Conforme à un scénario d'utilisation
- Facilite la connexion entre composants

Analyse

- Réduction importante de la combinatoire (connexions entre ports)
- Contrainte de connexion trop importante (peu de solutions)

Plan

- 1 Aide à la construction d'architectures valides satisfaisant un objectif fonctionnel
- 2 Validité d'une architecture
- 3 Construction d'architectures potentiellement valides satisfaisant un objectif fonctionnel**
 - Quelles interfaces connecter ?
 - Où connecter les interfaces ?**
 - Quand arrêter les connexions ?
- 4 Conclusion et perspectives

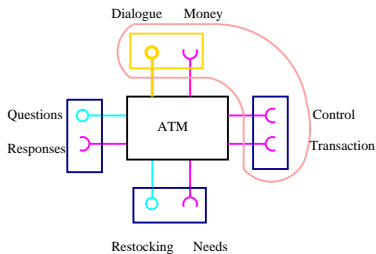
Où connecter les interfaces ?

- A un même composant
- A des composants différents

Un nouveau concept : port composite

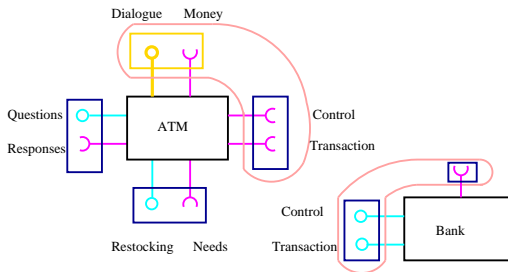
- Regroupe un ensemble de ports primitifs et composites
- Exprime la structuration d'un scénario d'utilisation en sous-scénarios d'utilisations
- Relaxe la contrainte de connexion à un même port
- Définit où connecter *in fine* les interfaces

Un nouveau concept : port composite



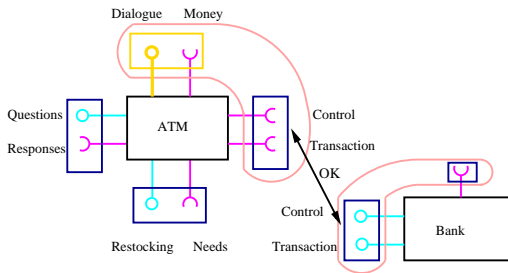
Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Un nouveau concept : port composite



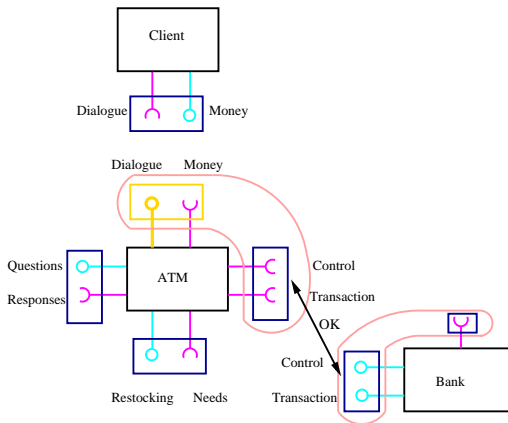
Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Un nouveau concept : port composite



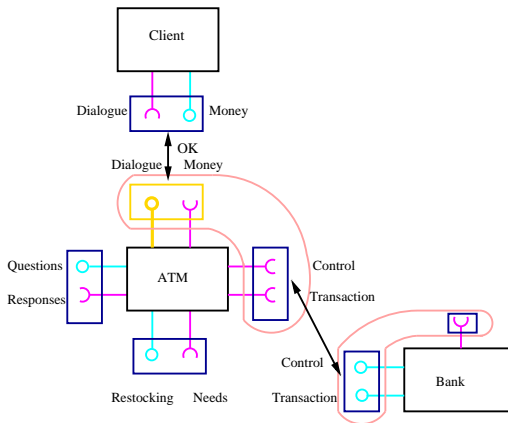
Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Un nouveau concept : port composite



Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Un nouveau concept : port composite



Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Plan

- 1 Aide à la construction d'architectures valides satisfaisant un objectif fonctionnel
- 2 Validité d'une architecture
- 3 Construction d'architectures potentiellement valides satisfaisant un objectif fonctionnel**
 - Quelles interfaces connecter ?
 - Où connecter les interfaces ?
 - Quand arrêter les connexions ?**
- 4 Conclusion et perspectives

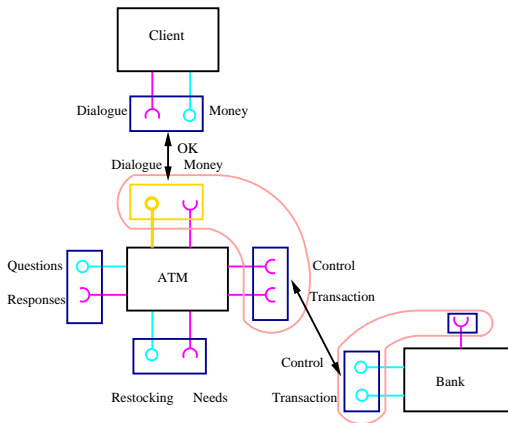
Définition

- 1 Une architecture est **quasi-valide** si tous ses ports composites sont cohérents
- 2 Un port composite est **cohérent** si ses ports primitifs (directs et indirects) sont tous connectés ou tous déconnectés
- 3 Une architecture A est quasi-valide et satisfait son objectif fonctionnel si
 - Quasi-valide(A)
 - Tout port primitif représentant l'objectif fonctionnel est connecté

Propriété

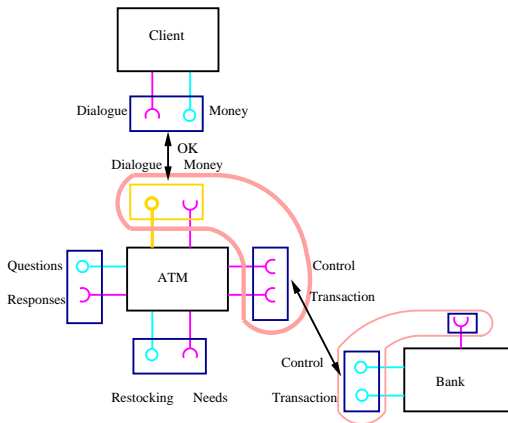
- Une architecture A construite à l'aide de ports, vérifie :
 - Valide(A) \Rightarrow Quasi-valide(A)

Propriété de quasi-validité



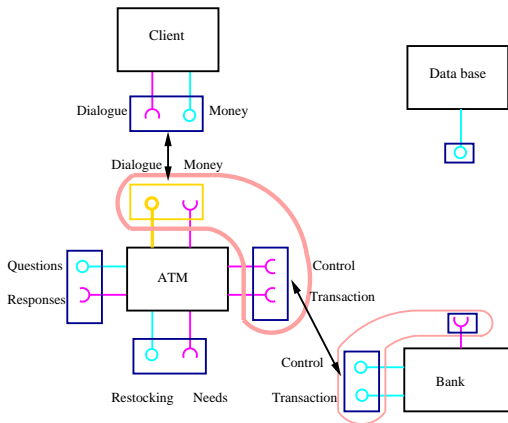
Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Propriété de quasi-validité



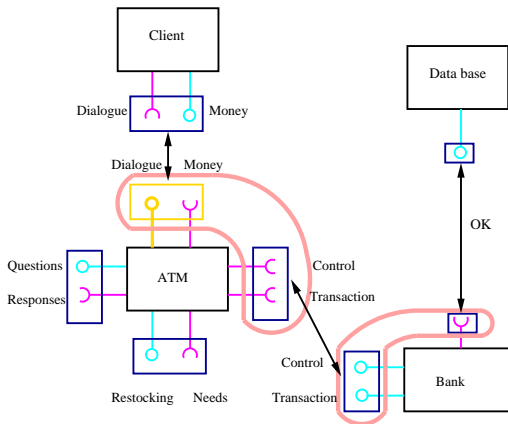
Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Propriété de quasi-validité



Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Propriété de quasi-validité



Money	getMoney getTicket
Dialogue	connect cancel disconnect balance withdrawal OF
Control	checkBalance getStockInformation
Transaction	clientIdentify withdrawal balance
Query	request
Restocking	setMoney
Needs	askMoney

Bilan

Résultats

- Quasi-validité : permet de trouver un algorithme systématique de construction d'architectures
- Port : réduction de la combinatoire
 - heuristiques : exploration complète de l'espace de recherche pour y trouver des architectures minimales

3 niveaux d'assistance à l'architecte

- Non automatique : aspect intuitif
 - Port : concept de plus haut niveau d'abstraction que l'interface
 - Assemblage entre ports
- Semi-automatique
 - Assistance dans le choix des composants
- Automatique : proposition d'architectures

Conclusion

- Réduction de manière très forte du nombre d'architectures candidates à la validité
- Construction d'architectures potentiellement valides satisfaisant un objectif fonctionnel
- Propriété de quasi validité
 - Nouveau concept : port composite
- Implémentation dans le modèle de composants Fractal

Perspectives

- Composition hiérarchique
- Génération des ports à partir des protocoles
- Substitution dynamique

Conclusion

- Réduction de manière très forte du nombre d'architectures candidates à la validité
- Construction d'architectures potentiellement valides satisfaisant un objectif fonctionnel
- Propriété de quasi validité
 - Nouveau concept : port composite
- Implémentation dans le modèle de composants Fractal

Perspectives

- Composition hiérarchique
- Génération des ports à partir des protocoles
- Substitution dynamique