

Déploiement adaptatif des composants dans les sessions collaboratives

Emir HAMMAMI, **Thierry VILLEMUR**

{ehammami, **villemur**}@laas.fr

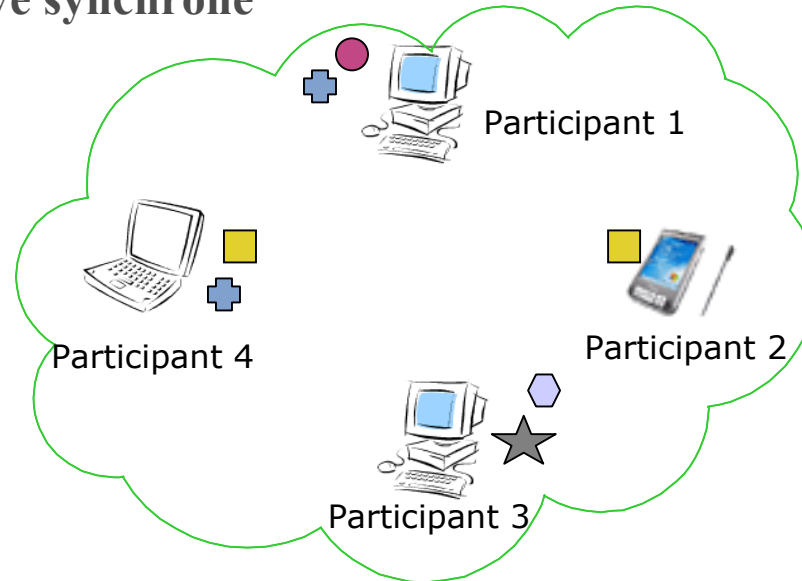


LAAS-CNRS
7, avenue du Colonel Roche
31077 Toulouse Cedex 4
France

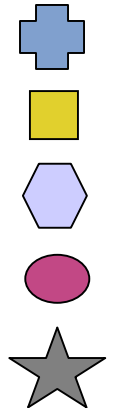
29 Août au 1^{er} Septembre 2005, Gatineau, Canada.

Contexte

Session collaborative synchrone



Composants



Déploiement des composants

L'ensemble des activités permettant de rendre les composants disponibles sur les sites des participants

Plan

1. Besoin du déploiement dynamique
 - Adaptabilité au contexte
2. Approche de déploiement
 - Les étapes du déploiement
 - Modèle de session
 - Modèle de composant
 - Modèle de site
 - Algorithme de déploiement
3. Réalisation
 - Plate-forme de déploiement
 - Architecture proposée
4. Conclusion et Travaux Futurs

Pourquoi le déploiement dynamique sensible au contexte ?

- Contextes variés : équipements hétérogènes
- Ressources limitées : espace disque, mémoire, capacité de traitement, ...
- Évolution de la session : entrée/sortie de participants, ...
- Déploiement à large échelle

Adaptabilité au contexte

Déploiement sensible au contexte

L'ensemble des activités permettant de rendre les composants disponibles sur les sites des participants **en tenant en compte des informations de contexte**

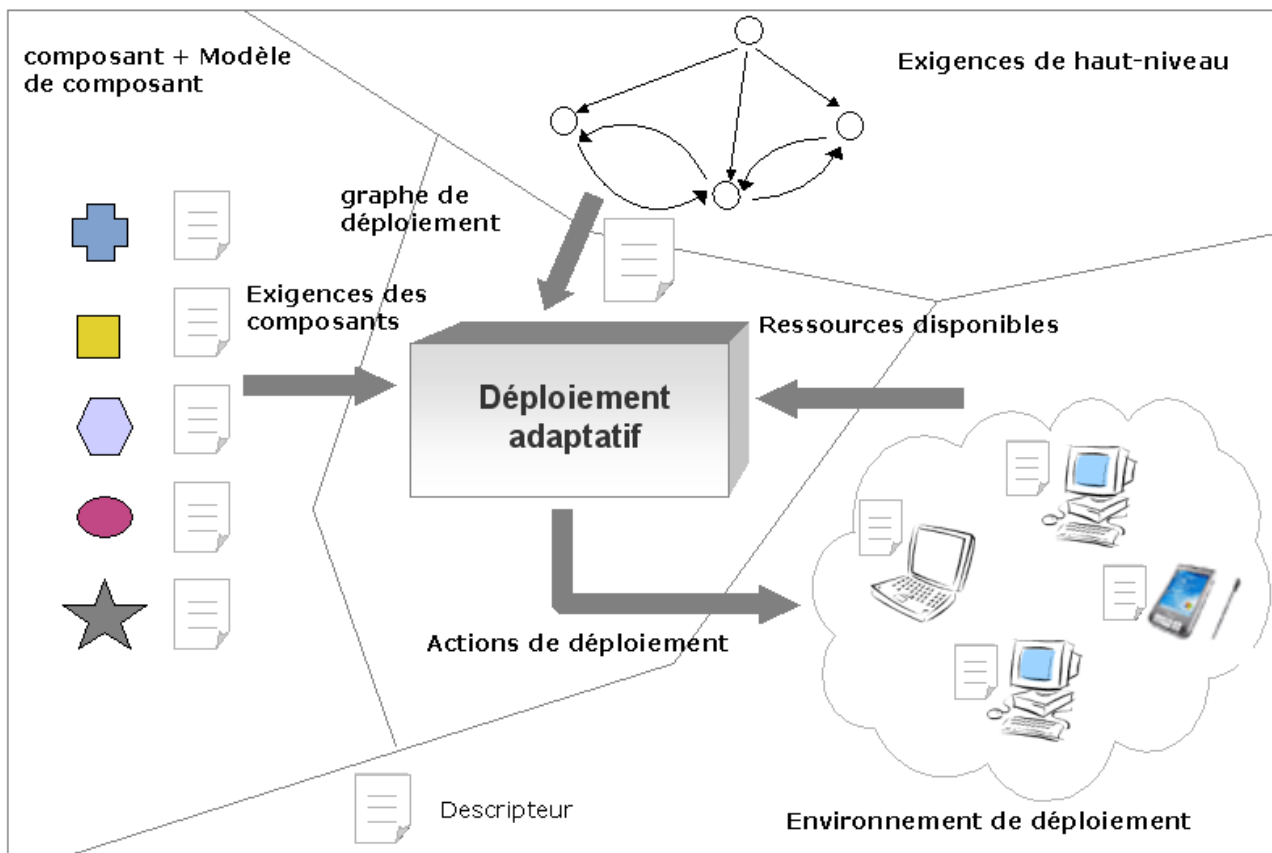
Contexte local : lié à l'équipement de déploiement

- Ressources offertes
- Qualité de Service offertes

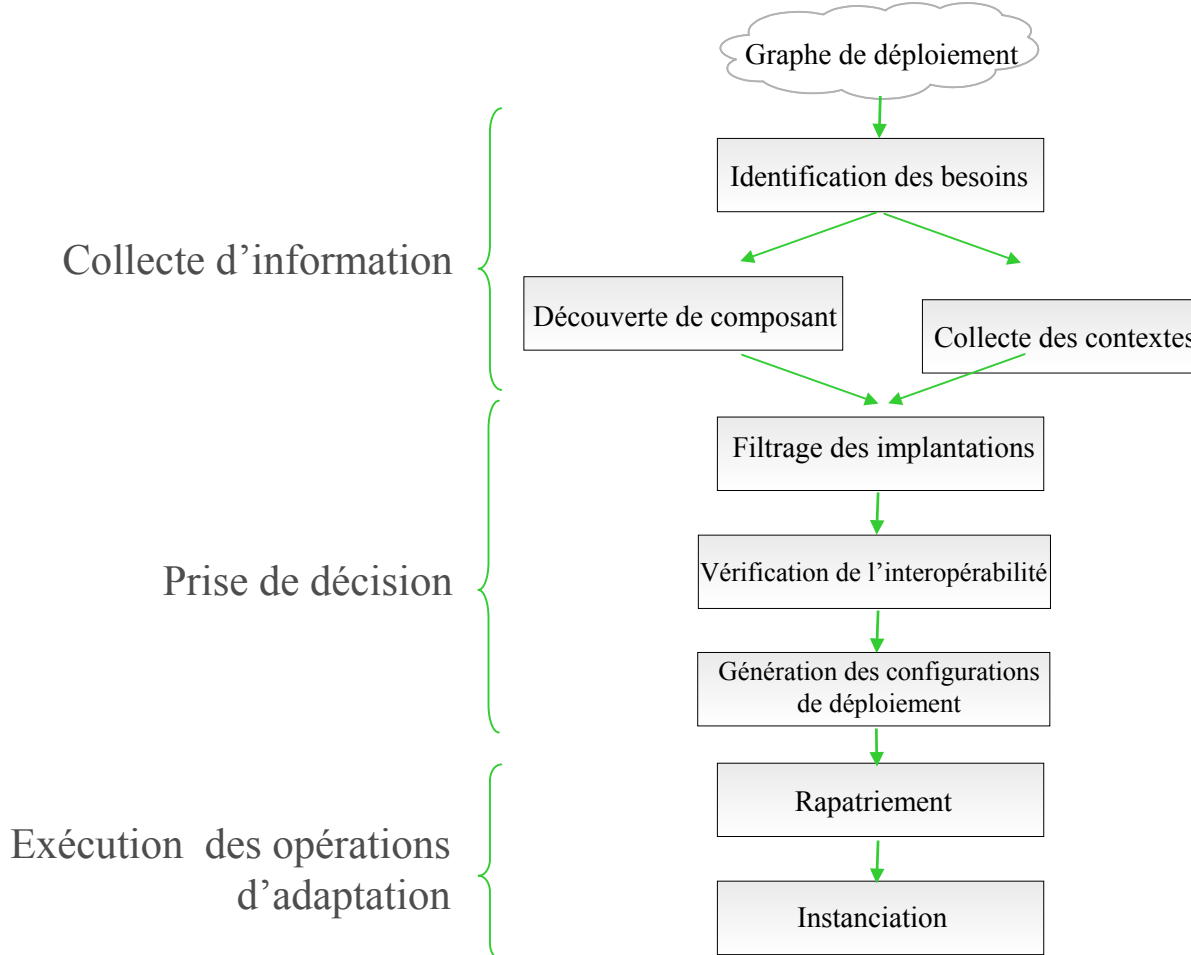
Contexte distribué : pour garantir l'interopérabilité entre composants

Contexte utilisateur : préférences de l'utilisateur

Notre approche



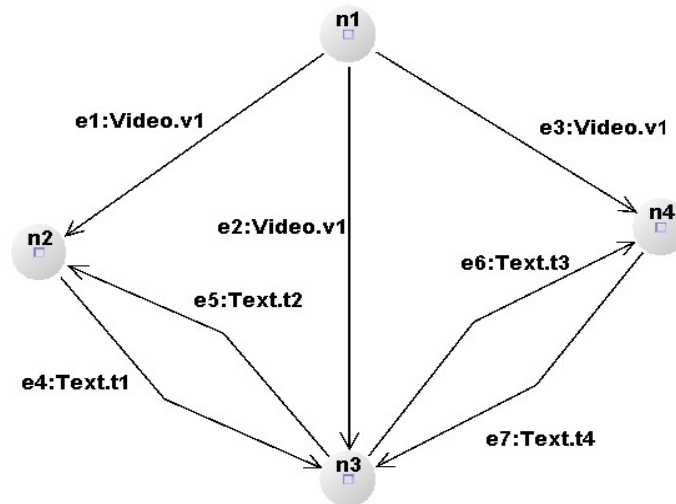
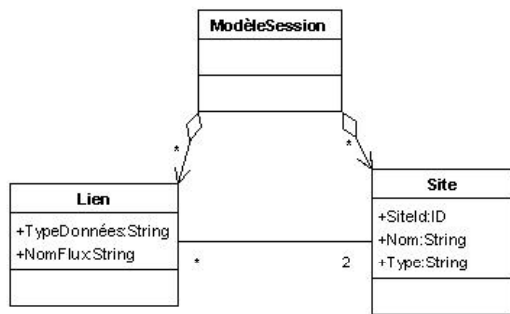
Les étapes du déploiement adaptatif



Modèles pour automatiser le déploiement

- Modèle de session
- Modèle de composant
- Modèle de site

Modèle de session



```
<DeploymentGraph id="g-0003">
  <Nodes>
    <Node>n1</Node>
    <Node>n2</Node>
    <Node>n3</Node>
    <Node>n4</Node>
  </Nodes>
  <Edges>
    <Edge id="e1" src="n1" tgt="n2">
      <DataType>Video</DataType>
      <Flowname>v1</Flowname>
    </Edge>
    <Edge id="e2" src="n1" tgt="n3">
      <DataType>Video</DataType>
      <Flowname>v1</Flowname>
    </Edge>
    <Edge id="e3" src="n1" tgt="n4">
      <DataType>Video</DataType>
      <Flowname>v1</Flowname>
    </Edge>
    <Edge id="e4" src="n2" tgt="n3">
      <DataType>Text</DataType>
      <Flowname>t1</Flowname>
    </Edge>
    <Edge id="e5" src="n3" tgt="n2">
      <DataType>Text</DataType>
      <Flowname>t2</Flowname>
    </Edge>
    <Edge id="e6" src="n3" tgt="n4">
      <DataType>Text</DataType>
      <Flowname>t3</Flowname>
    </Edge>
    <Edge id="e7" src="n4" tgt="n3">
      <DataType>Text</DataType>
      <Flowname>t4</Flowname>
    </Edge>
  </Edges>
</DeploymentGraph>
```

Avantages du modèle

- Représentation à haut niveau
- Modèle programmable pour être interprété, et pris en compte par une couche logicielle de gestion de déploiement

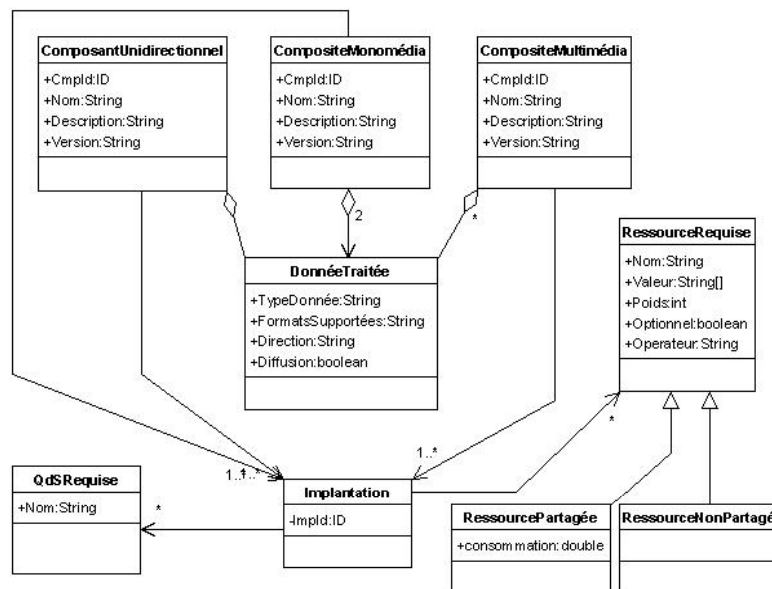
Modèle de composant

Classes de composants :

- 1- Monolithique monomédia unidirectionnel
- 2- Composite monomédia producteur consommateur
- 3- Composite multimédia

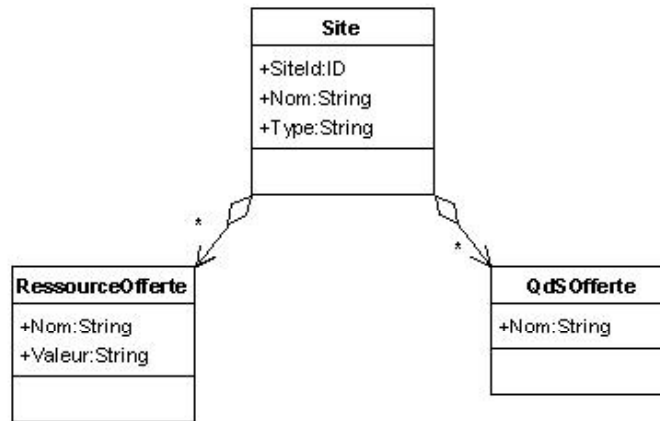
Avantages du modèle

- Couvrir tous les types de composants
- Guider le choix des composants à déployer



```
<monolithicunidirectional>
<id>md5-0e6ac527ce2acddf2f339d5020c16de</id>
<name>DialogProd</name>
<dataprocess>
  <dataprocess>
    <datatype>Text</datatype>
    <supportedformat>
      <format>ascii</format>
    </supportedformat>
    <direction>producer</direction>
    <multicast>true</multicast>
    <sameflow>true</sameflow>
  </dataprocess>
</dataprocess>
<implementations>
  <implementation>
    <id>md5-8566f397f204433a2b504d7c951820b</id>
    <name>impl1p</name>
    <version>version2</version>
    <codefile>z:\prog-prod.jar</codefile>
    <require qos>persistency,fiability</required qos>
    <notshare requiredresource>
      <resource>
        <name>os</name>
        <type>Text</type>
        <value>Windows</value>
        <operator>superior</operator>
        <optional>true</optional>
        <weight>2</weight>
      </resource>
      <resource>
        <name>cpu</name>
        <type>Integer</type>
        <value>25</value>
        <operator>superior</operator>
        <optional>true</optional>
        <weight>2</weight>
      </resource>
    </notshare requiredresource>
    <share requiredresource>
      <resource>
        <name>mem</name>
        <type>Double</type>
        <value>32.0</value>
        <operator>inferior</operator>
        <optional>true</optional>
        <consumption>40</consumption>
        <weight>2</weight>
      </resource>
    </share requiredresource>
  </implementation>
</implementations>
</monolithicunidirectional>
```

Modèle de site



```
<deploymentcontext>
  <id>md5-e20d37a5d7fcc4c35be6fc18a8e71bfa</id>
  <name>Toulouse</name>
  <type>PC</type>
  <address>195.83.90.82</address>
  <relevantcontext>
    <offeredresources>
      <resource>
        <name>os</name>
        <type>Text</type>
        <value>Linux</value>
      </resource>
      <resource>
        <name>mem</name>
        <type>integer</type>
        <value>128</value>
      </resource>
    </offeredresources>
    <offeredqos>
      <qos>
        <name>persistence</name>
      </qos>
      <qos>
        <name>security</name>
      </qos>
    </offeredqos>
  </relevantcontext>
</deploymentcontext>
```

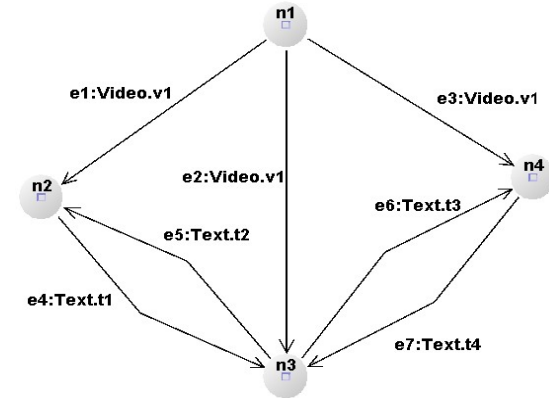
En combinant ces informations avec les exigences des composants, il est possible de vérifier si un composant peut fonctionner en local correctement ou non

⇒ donc d'autoriser ou de refuser son déploiement.

Algorithme de déploiement adaptatif

Identification des besoins

A partir du graphe de session, on détermine :



- L'ensemble des types de données et le sens de communication (production ou consommation)

Pour n3, $\{$ b1={Vidéo, Cons.},
b2={Texte, Prod.},
b3={Texte, Cons.} $\}$

- Les nœuds voisins avec lesquelles les composants vont communiquer

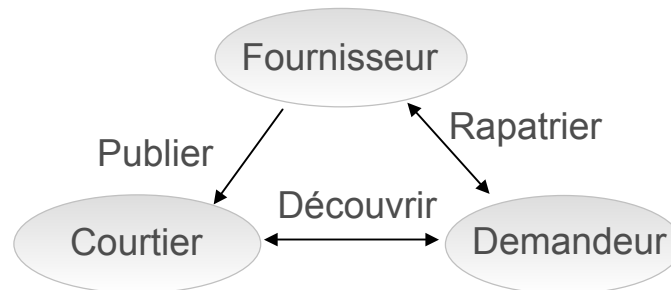
$\{b1 \rightarrow \{n1\},$
 $b2 \rightarrow \{n2, n4\},$
 $b3 \rightarrow \{n2, n4\}\}.$

Découverte des composants

Découverte distribuée avec recherche par mots clés

Critères de découverte : Type de données + sens de communication

Résultat : Liste de descripteurs des composants qui répondent aux besoins d'un participant à la session

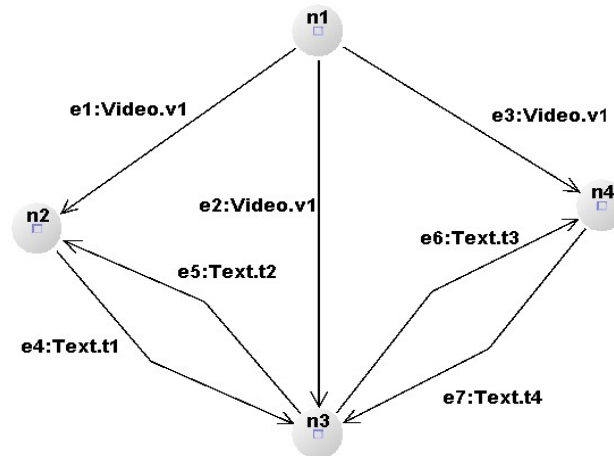


Collecte des contextes des voisins

Permet de récupérer la liste des composants (des implantations) déjà déployées sur les sites voisins.

Préparation pour l'étape de vérification de l'interopérabilité.

Exemple : Le nœud n3 récupère les informations issues des nœuds n1, n2 et n4.



Sélection des implantations

Comparaison entre :

- les ressources et les paramètres de QdS requis par une implantation et
- les ressources et les paramètres de QdS offerts par un site.

1. *Pour chaque besoin b_i identifié dans la première phase faire :*
2. *Pour chaque descripteur de composant d_j vérifiant b_i faire :*
3. *Pour chaque implantation I_k contenue dans d_j faire :*
4. *si incompatible (I_k , descripteur de site)*
5. *alors éliminer I_k du descripteur d_j*
6. *Fin*
7. *Fin*
8. *Fin*

Seules les **implantations compatibles avec le contexte local** sont gardées, les autres sont éliminées.

Vérification de l'interopérabilité

Pour **chaque implantation I**, l'algorithme détermine, les **nœuds voisins** qui possèdent des **composant interopérables** avec cette implantation.

1. *Pour chaque besoin b_i identifié lors de la première phase :*
2. *Pour chaque implantation I_j vérifiant b_i faire :*
3. *Pour chaque nœud voisin n_k restreint à b_i faire :*
4. *si $\text{interopérable}(I_j, n_k, b_i^{-1})$*
5. *alors ajouter n_k à la liste des sites interopérables de I_j*
6. *Fin*
7. *Fin*
8. *Fin*

b_i^{-1} exprime le besoin opposé. Si $b_1 = \{\text{Vidéo, Prod.}\}$ alors $b_1^{-1} = \{\text{Vidéo, Cons.}\}$.

Par exemple, pour n_1 , les implantations candidates sont $\{I_1, I_2, I_3, I_4\}$

Résultat de cette étape: $I_1 \rightarrow \{n_2, n_3, n_4\}$; $I_2 \rightarrow \{n_3\}$; $I_3 \rightarrow \{n_3, n_4\}$; $I_4 \rightarrow \{\}$.

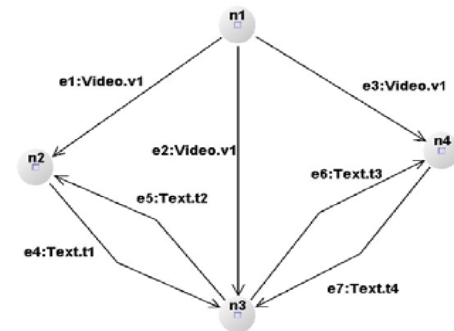
Génération des configurations de déploiement

- L'algorithme élimine les implantations qui ne sont pas interopérables même avec au moins un voisin (l'implantation I4 est éliminée)
- Génère toutes les **combinaisons possibles des composants** vérifiant les critères de **compatibilité locale** et **d'interopérabilité** (Configurations 1 à 7)
- Vérifie pour chaque configuration si elle satisfait tous les nœuds voisins ou pas (Configurations valides : 1, 4, 5 et 7)

Une configuration de déploiement est valide si elle couvre la solution sur un nœud particulier.

Pour n1,

	Configuration	n2	n3	n4
1	{I1}	✓	✓	✓
2	{I2}		✓	
3	{I3}		✓	✓
4	{I1, I2}	✓	✓	✓
5	{I1, I3}	✓	✓	✓
6	{I2, I3}		✓	✓
7	{I1, I2, I3}	✓	✓	✓



Rapatriement

Transfert des implantations depuis les **sites fournisseurs** de composants vers le **site de déploiement**

Le choix d'une configuration valide est influencé par :

-Les implantations déjà déployées

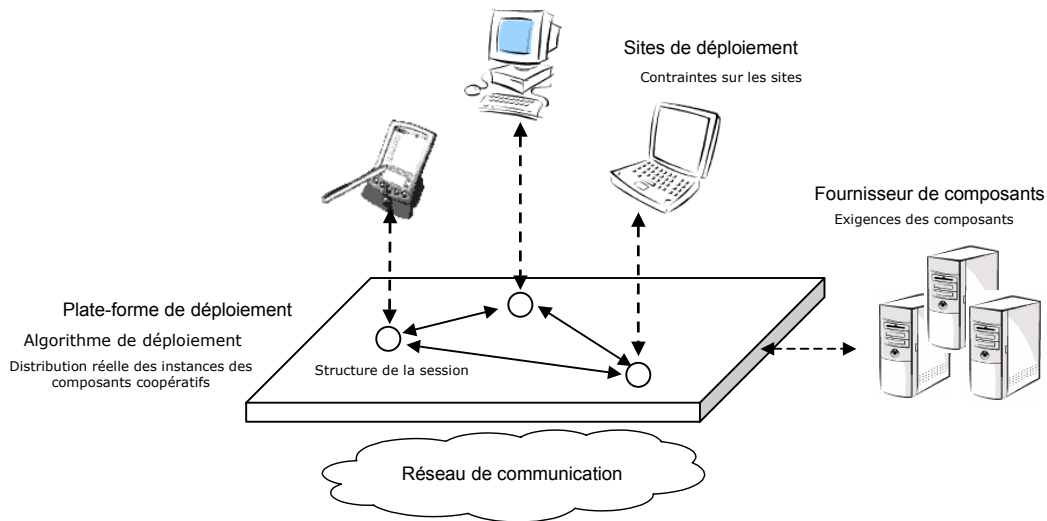
-Minimiser le nombre de rapatriement à faire en favorisant les composants composites multimédia par rapport à ceux des autres classes et en favorisant les composants composites monomédia par rapport à ceux de la classe monolithique

Instanciación

Le nombre d'instanciations est déterminé par la capacité du composant à supporter plusieurs flux de données d'un même type ou non

- Un composant producteur (resp. consommateur) multiple ne sera instancié qu'une seule fois.
- Un composant producteur (resp. consommateur) simple sera instancié une fois par flux, soit N fois pour N flux.

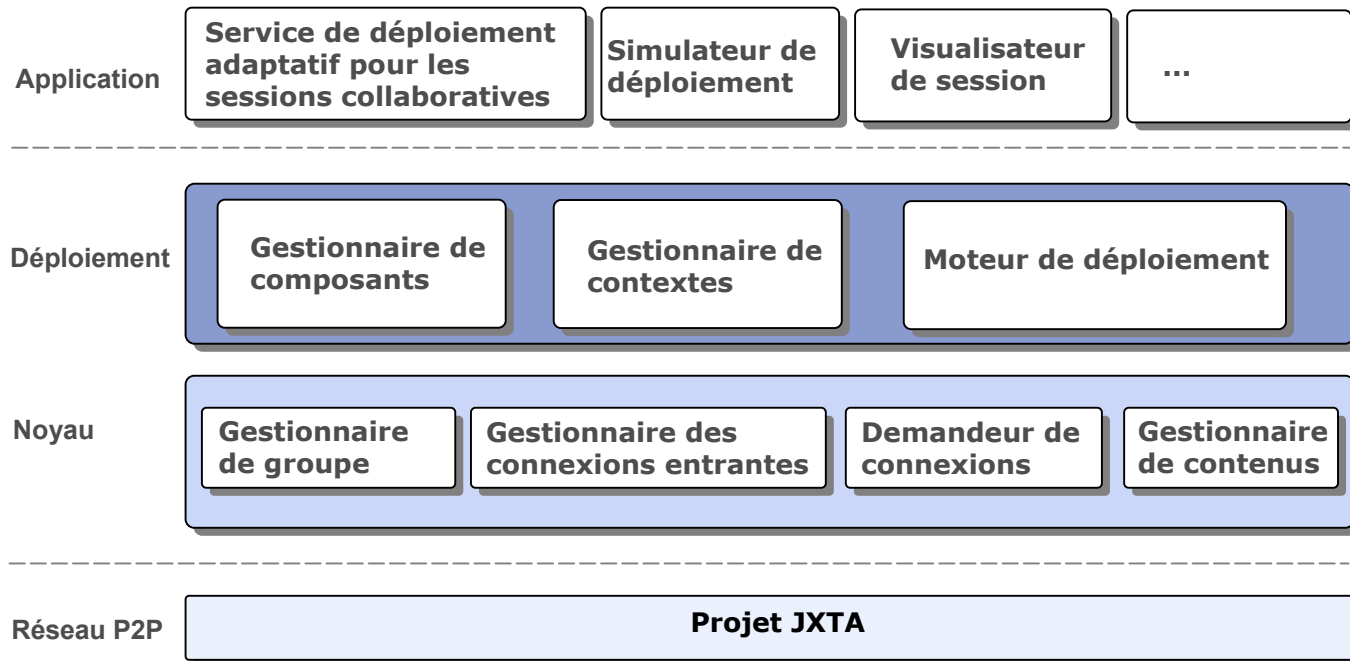
Plate-forme de déploiement



Objectifs

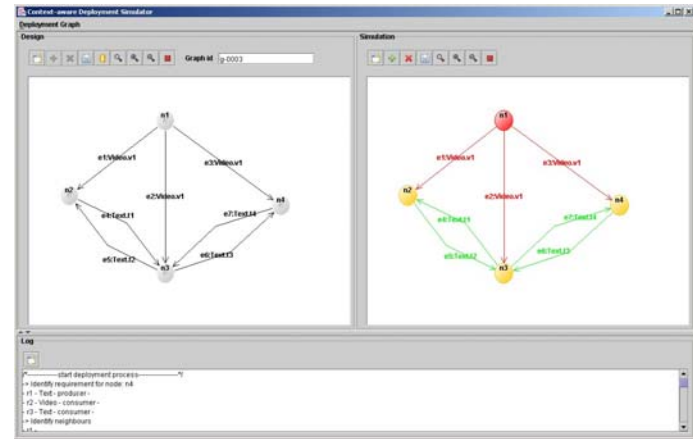
- ▶ Fournir une API générique permettant le déploiement des composants dans un environnement pair-à-pair
- ▶ Peut être adaptée à divers domaines d'application comme le travail collaboratif, ou les applications distribuées multi-composant
- ▶ Masque la complexité des traitements sous-jacents

Architecture de la plate-forme



Développement

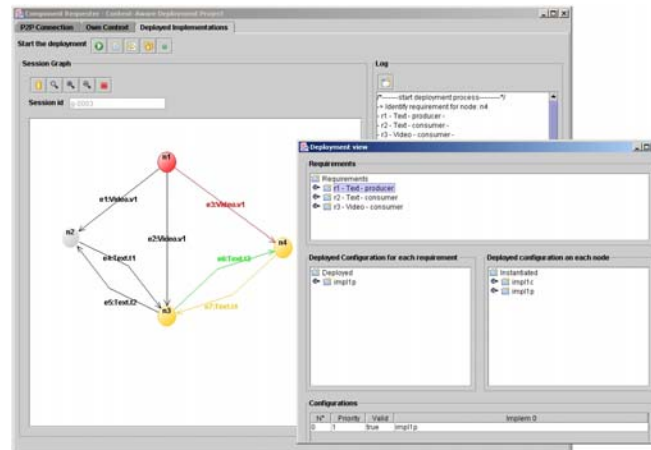
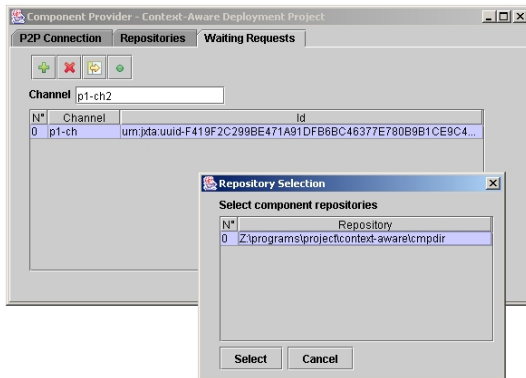
Simulateur de déploiement (local)



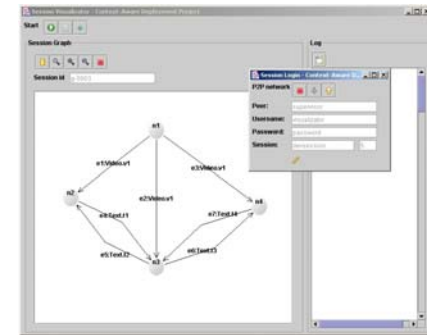
Service de déploiement (distribué)

Site de déploiement

Fournisseur de composants



Visualisateur de session



Conclusion

- Étude de la problématique du déploiement adaptatif
- Notre approche :
 - Modèles pour automatiser le déploiement
 - Algorithme qui supporte les étapes du déploiement
- API générique pour le déploiement
 - Indépendante de son domaine d'utilisation

Travaux futurs

- Validation des concepts présentés
- Études de complexité de l'algorithme
- Évaluations de performances, de fiabilité et de passage à l'échelle

Merci

