Formation AmbientComp

Stéphane Lavirotte, Jean-Yves Tigli, Gaëtan Rey,
Thibaut Gonnin, Gérald Rocher

Equipe SPARKS, Laboratoire I3S, UMR CNRS 7271, Université de Nice Sophia Antipolis,

Email : prenom.nom@univ-cotedazur.fr

Tutorial AmbientComp pour la composition locale dynamique

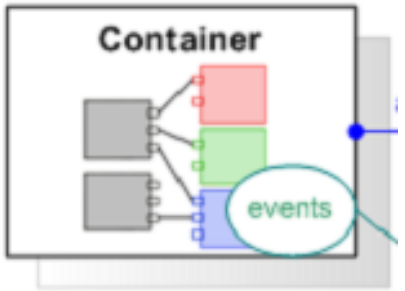
Modèle LCA, Conception de Beans

Intergiciel AmbientComp

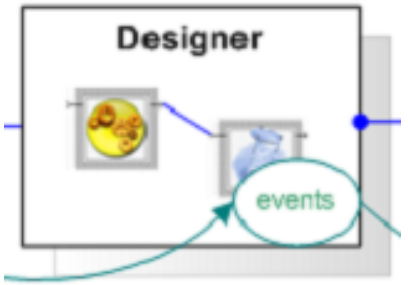
- Avantage :
  - Architecture orientée composant
  - Composition dynamique au runtime
- Modèle du Middleware (LCA)
  - Modèle de Bean AmbientComp (Propriété / Méthodes / Evénements)
  - Connecteurs AmbientComp
- Créer / Gérer votre application par assemblage de composants

Architecture de AmbientComp : Designers - Containers

- Conteneur(s)
  - Gestionnaire des assemblages de composants légers (composants & connecteurs)

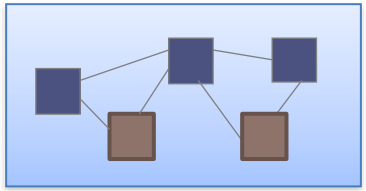


- Designer(s)
  - Outil de manipulation des assemblages en interaction avec les conteneurs
  - Ex. Designer Graphique, Designer Visuel, Designer Aspects d'Aspects d'Assemblage,…



Modèle LCA

- LCA : « *Lightweight Components Architecture* »

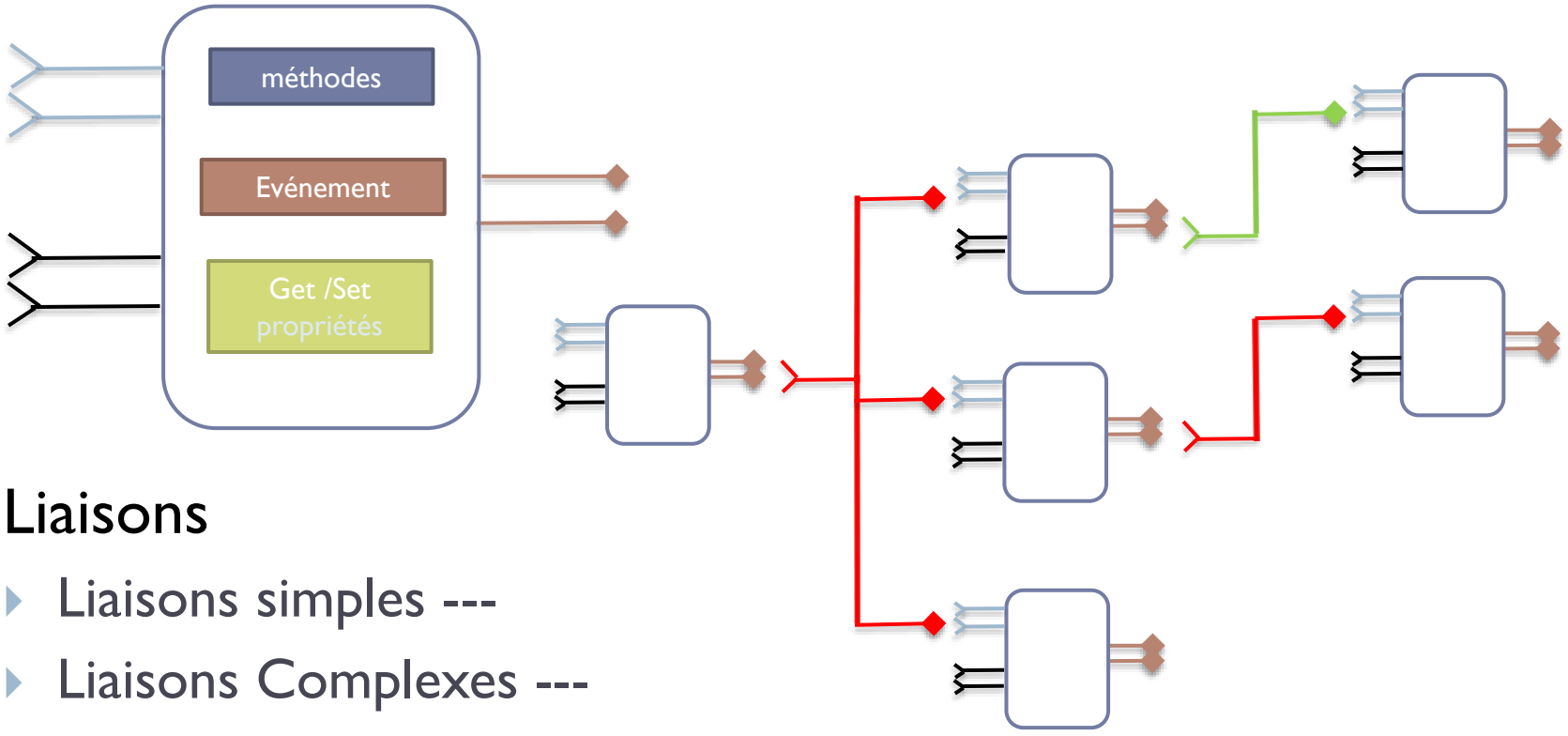


- L'Application est un Assemblage de Composants légers
- Composition par flots d'événements
- Nœud d'exécution et distribution explicite

- Proche de la notion d'orchestration

Des composants légers Bean

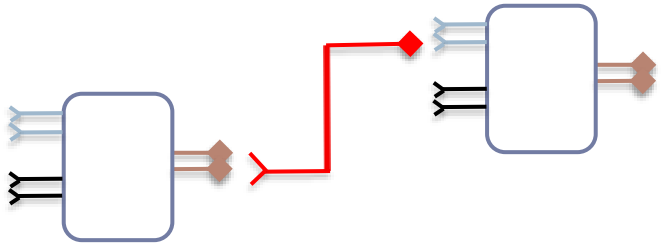
- Composant Bean



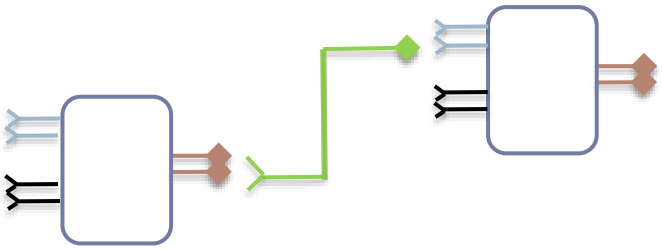
- Liaisons
  - Liaisons simples ---
  - Liaisons Complexes ---

Modèles de connecteurs : Event et Event Complexe

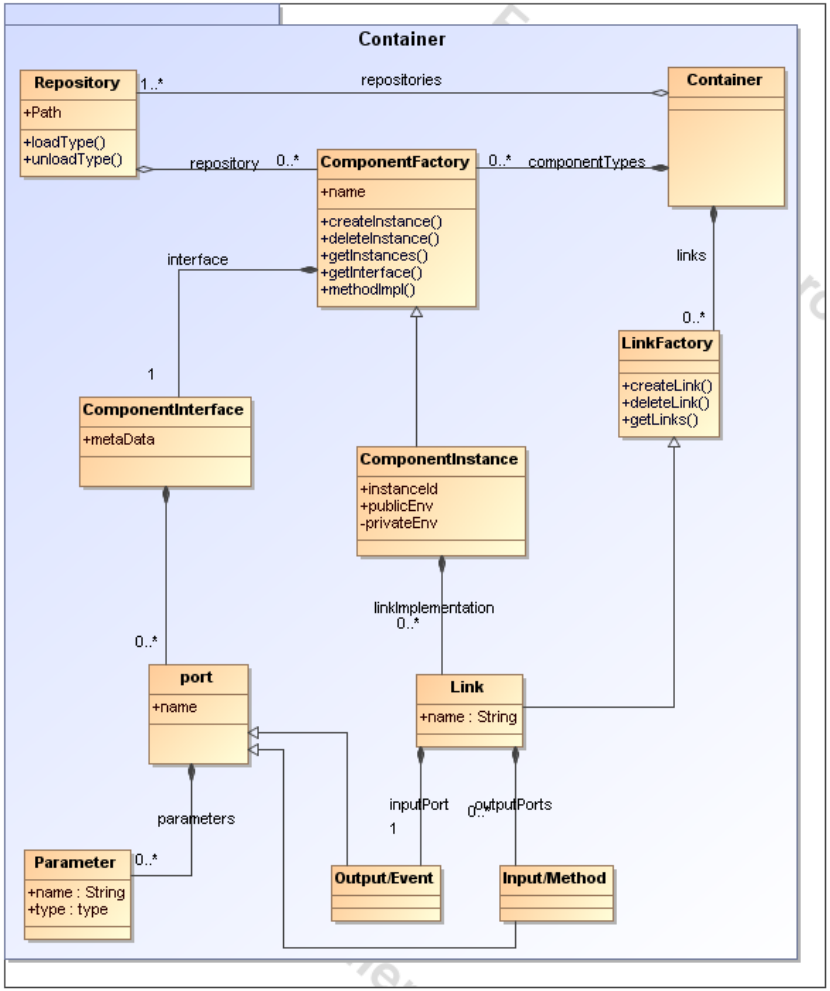
- Connecteur Event Simple
  - C1.Event (param) => C2.Methode (param)



- Connecteur Event Complexe
  - C1.Event (param) => C2.Methode (C1.GetProp())



Modèle LCA



- Implémentations courantes du modèle LCA:
  - C# .NET
    - Windows
    - RTLinux (expérimentale)
    - Android (expérimentale)
- Implémentations non maintenues
  - J2SE
  - J2ME
  - RTLinux

Exemple de Bean .Net

- Implémentation d'un Composant AmbientComp
  - Créer un Bean en .NET C#

Attributs personnalisé (exemple catégorie de rangement dans l'interface)

```
using System;
using System.ComponentModel;
using WComp.Beans;

namespace WComp.Beans.Tuto
{
    /// <summary>
    /// Description de MyBean
    /// </summary>
    [Bean(Category="Tuto")]

    public class MyBean
    {
```

Nom du Bean

Propriété de Bean .Net

- Propriété

```
...
// Variable interne de sauvegarde de la propriété
private int myprop = 0;

//Meta donnée : valeur par défaut propriété
[DefaultValue(0)]

// Déclaration de la propriété : public <type> Nom
public int Myprop
{
     get
     {
          return myprop;
     }

     set
     {
          if (myprop > 0)
          {
               MyEvent(myprop)
          }
          // stockage de la valeur dans la variable interne
          myprop = value;
     }
}
...
```

Propriété

Méthode de Bean .Net

- Méthode

Méthode

```
// méthode
public void MyStep(int val1, int val2)
{
    if (myprop >= val1)
    {
        myprop = val2;
    }
    else
    {
        myprop++;
    }
}
```


Stéphane Lavirotte, Jean-Yves Tigli, Gaëtan Rey – prenom.nom@unice.fr

Evénement de Bean .Net

- Evénement

Définition du delegate: type d'événement

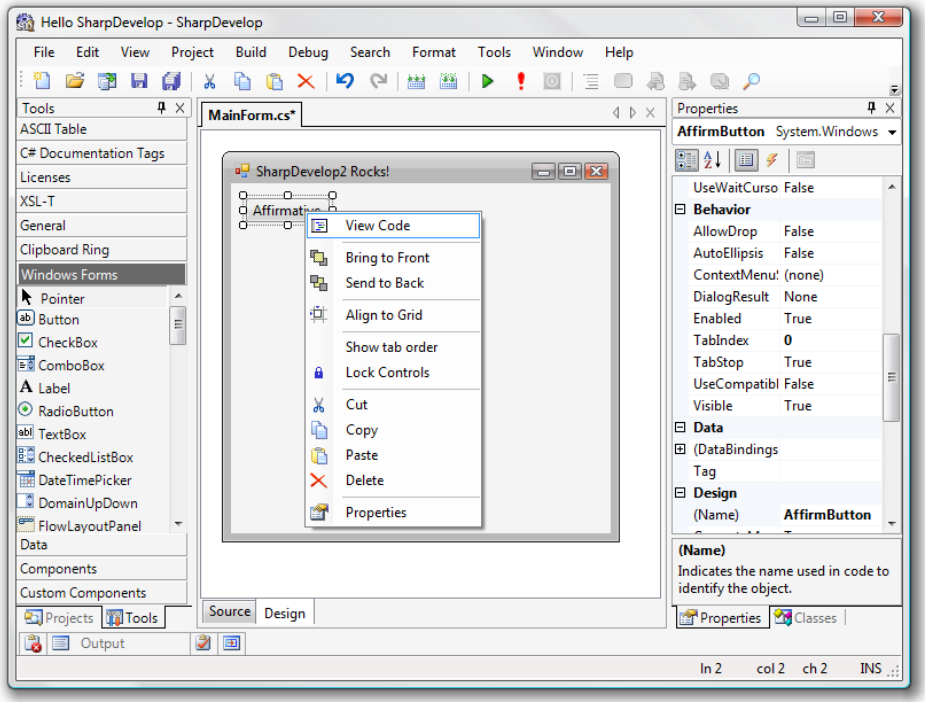
Définition de l'événement

Utilisation de l'événement dans le Bean

```
...
/// Le délégate correspondant à la signature de l'événement
///  assimilable à la définition d'un type
public delegate void IntValueEventHandler(int val);

/// La déclaration de l'événement lui-même son nom ici est "PropertyChanged"
/// Ce sera le nom visible dans l'interface du composant Bean
public event IntValueEventHandler PropertyChanged;



/// Fonction appelée dans le code du bean pour vérifier la non nullité
///  (vérifie qu'une méthode est "branchée" sur cet événement)
private void MyEvent(int i) {
   if (PropertyChanged != null)
      PropertyChanged(i);
}
...
```

Outils : Dans SharpDevelop (Addon)

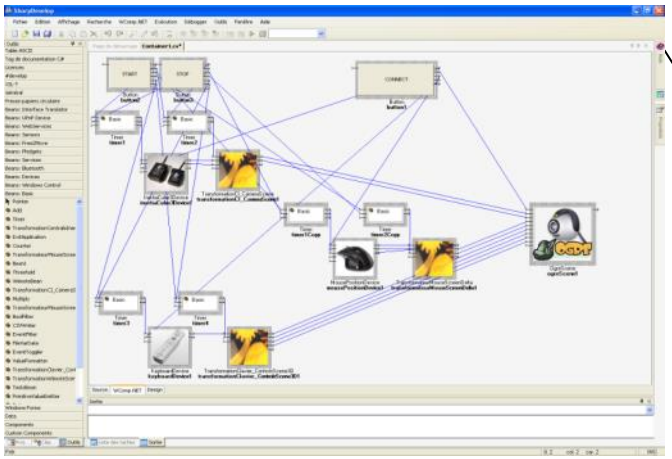
- SharpDevelop (SD) est un IDE basé sur .NET qui offre un environnement de développement de qualité comparable à VisualStudio
- SharpDevelop
  - Open Source
  - Licence L-GPL



http://www.icsharpcode.net/opensource/sd/

Différents designers dans SharpDevelop

- Designer visuel (modification graphique de l'assemblage de composant)
- Designer de code statique (projection en code source)



Représentation graphique de l'assemblage

- Designer graphique (modification de l'interface graphique)
- …

Mise en œuvre de LCA sous AmbientComp

- Installations
  - SharpDevelop 3.2
  - AmbientComp 3.2
- Réalisations:
  - Création d'un assemblage avec événements simples et complexes
  - Création d'un composant Bean
  - Création d'un assemblage utilisant le composant créé

Annexe 1

Delegate et Event en C# .NET

Annexe 1 : Delegate et Event en C# .Net

- Presque tout le monde connaît les événements.
- A chaque fois que vous cliquez sur un bouton dans une application Windows vous déclenchez un événement.
- Ici, le but est de créer des événements qui vont être propres à votre application. C'est-à-dire déclencher une action dans votre application lorsque quelque chose se produit et informer les objets abonnés à cet événement.

Annexe 1 : Delegate et Event en C# .Net

- Notre événement va nous servir à afficher un message toutes les secondes. Nous allons déclarer la classe comme ceci :

```
namespace TutoEvent
{
    /// <summary>
    /// A la responsabilité de contenir le text de l'évenement et de le rendre accessible

    /// </summary>
    public class GenerateTextEventArgs : EventArgs
    {
        private string myEventText = null;

        public GenerateTextEventArgs(string theEventText)
        {
            if (theEventText != null)
                myEventText = theEventText;
        }

        public string EventText
        {
            get { return this.myEventText; }
        }
    }
}
```

Annexe 1 : Delegate et Event en C# .Net

- http://freddyboy.developpez.com/dotnet/articles/events/

- Il n'est pas possible de parler des événements sans parler des delegate.
  - Un delegate est un objet qui permet d'appeler une fonction ou une série de fonction.
  - Un delegate est similaire aux pointeurs de fonctions du C/C++.

- Une variable delegate va permettre d'exécuter une fonction ou plusieurs fonctions.

- Pour cela le delegate va stocker des références sur des méthodes (que nous appellerons un gestionnaire d'évenéments ("Event handler")).

Annexe 1 : Delegate et Event en C# / .Net

- La signature des méthodes référencées devra respecter les règles suivantes :
  - retourner void
  - prendre comme premier paramètre un type object que nous appellerons généralement sender
  - prendre comme second paramètre un objet héritant de EventArgs, donc dans notre cas un objet GenerateTextEventArgs.

- Note : il est possible de voir des delegate déclarer autrement. Ici on montre simplement la méthode généralement utilisée pour les événements.

- Pour déclarer un delegate, nous utilisons la syntaxe suivante :
  - public delegate void TextGeneratedEventHandler (object sender, GenerateTextEventArgs e);

Event

- Il nous faut ensuite déclarer un objet event du type du delegate déclaré plus haut.

- Le mot clé event vous permet de spécifier un délégué à appeler lors de l'occurrence d'un certain événement dans votre code.

- Pour déclarer un event, nous utilisons la syntaxe suivante:
  - public event TextGeneratedEventHandler OnTextChanged;

Générer l'événement

- Pour générer un événement il suffit d'appeler son constructeur avec les paramètres éventuels comme ceci :
  - GenerateTextEventArgs e = new GenerateTextEventArgs("Compteur = " + i.ToString());
- Puis il nous reste à envoyer cet événement à tout le monde :
  - if (e != null) OnTextChanged(this,e);

Classe complète

```
using System;
using System.Threading;

namespace TutoEvent
{
    /// <summary>
    /// A la responsabilité d'envoyer un evenement GenerateTextEvent toutes les secondes

    /// </summary>
    public class GenerateText
    {
        /// <summary>
        /// Declare un delegate
        /// </summary>
        public delegate void TextGeneratedEventHandler(object sender, GenerateTextEventArgs e);
        /// <summary>
        /// Declare un evenement qui va contenir les informations que nous souhaitons envoyer
        /// </summary>
        public event TextGeneratedEventHandler OnTextChanged;

        public GenerateText(){}

        public void Start(int theNumber)
        {
            int i = 0;
            while (i  < theNumber)
            {
                GenerateTextEventArgs e = new GenerateTextEventArgs("Compteur = " + i.ToString());
                if (e != null) OnTextChanged(this,e);
                Thread.Sleep(1000);
                i++;
            }
        }
    }
}
```

Récupérer un événement dans un gestionnaire d'événements

- Un gestionnaire d'évenéments ("Event Handler") est la méthode qui va s'exécuter en réponse à l'événement.

- Un Event hanlder retourne normalement void et accepte 2 paramètres qui sont:
  - le sender : l'objet dans lequel l'événement s'est produit.
  - Un argument de type EventArgs qui contient les informations relatives à l'événement.

- Pour récupérer un événement, la première chose à faire est de se placer à l'écoute de cet événement. C'est là que le delegate que nous avons déclaré plus haut trouve toute son utilité.
- La syntaxe pour ajouter un gestionnaire d'évenéments ("Event handler") à l'écoute d'un événement est la suivante :
  - private GenerateText myTextGenerator = new GenerateText();
  - myTextGenerator.OnTextChanged+=new TutoEvent.GenerateText.TextGeneratedEventHandler(myTextGenerator_MonEvenement);