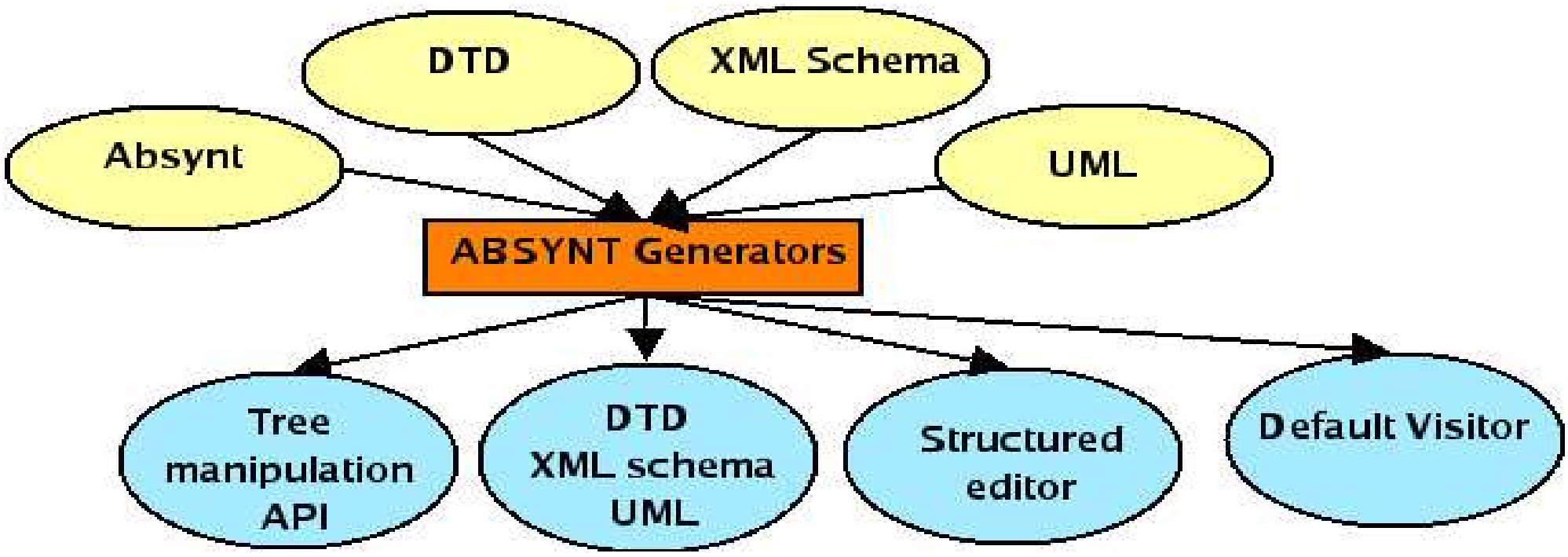
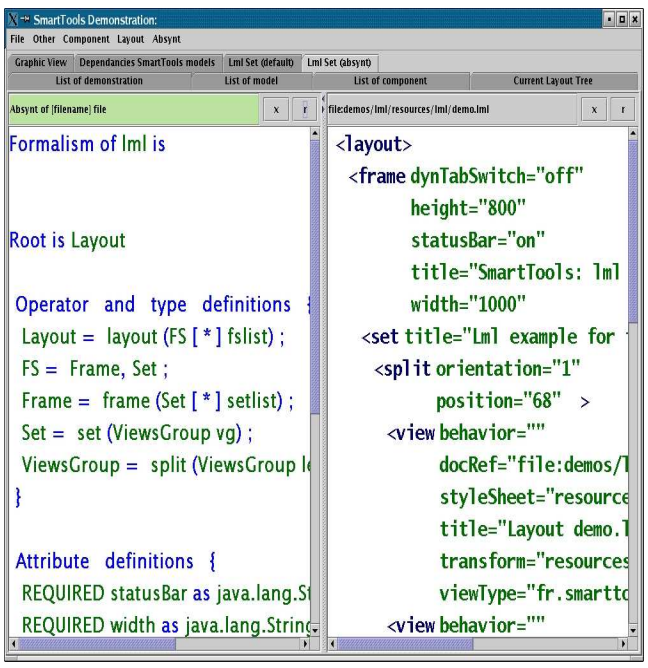
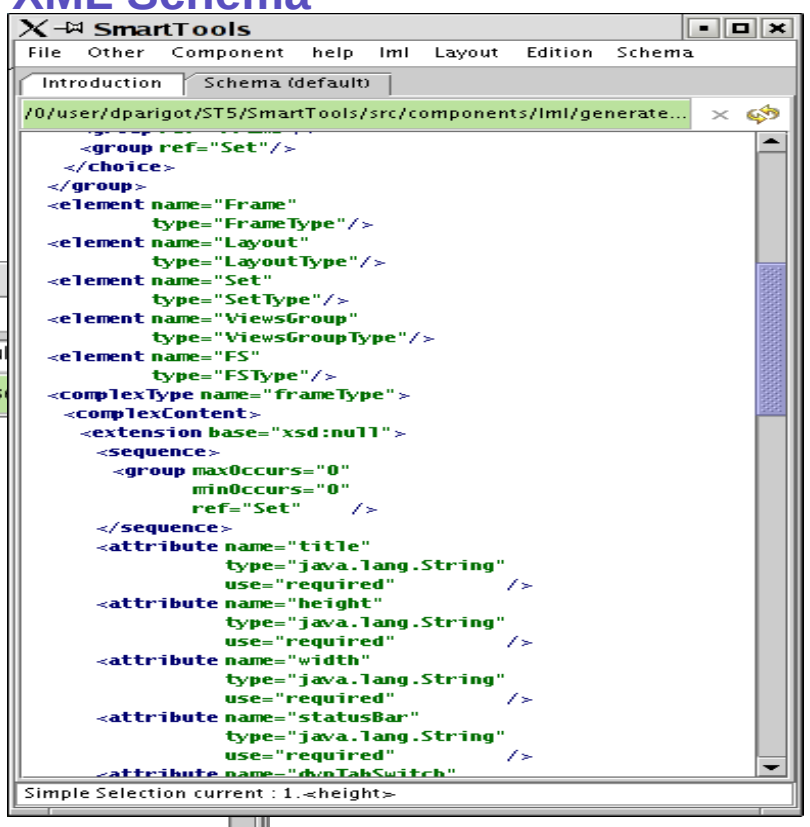
DSL [Absynt] pour définir des DSLs



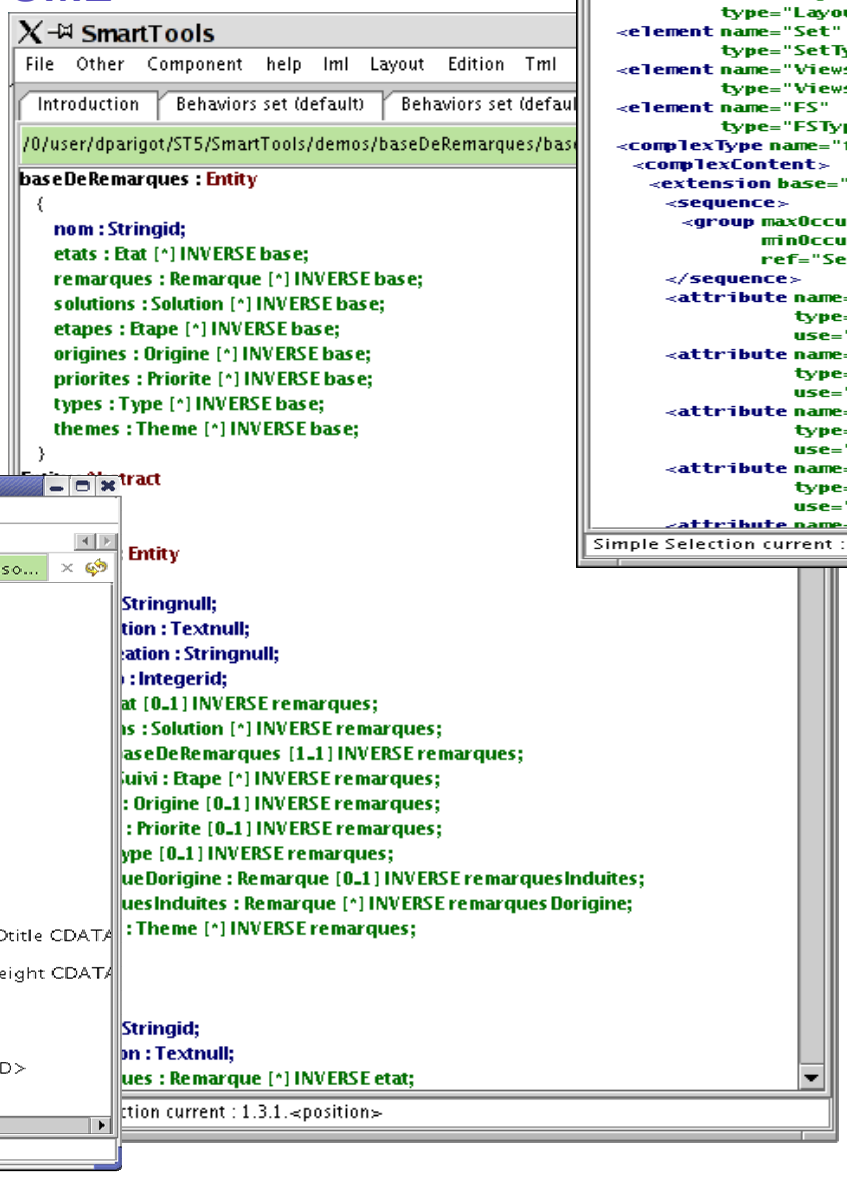
Absynt



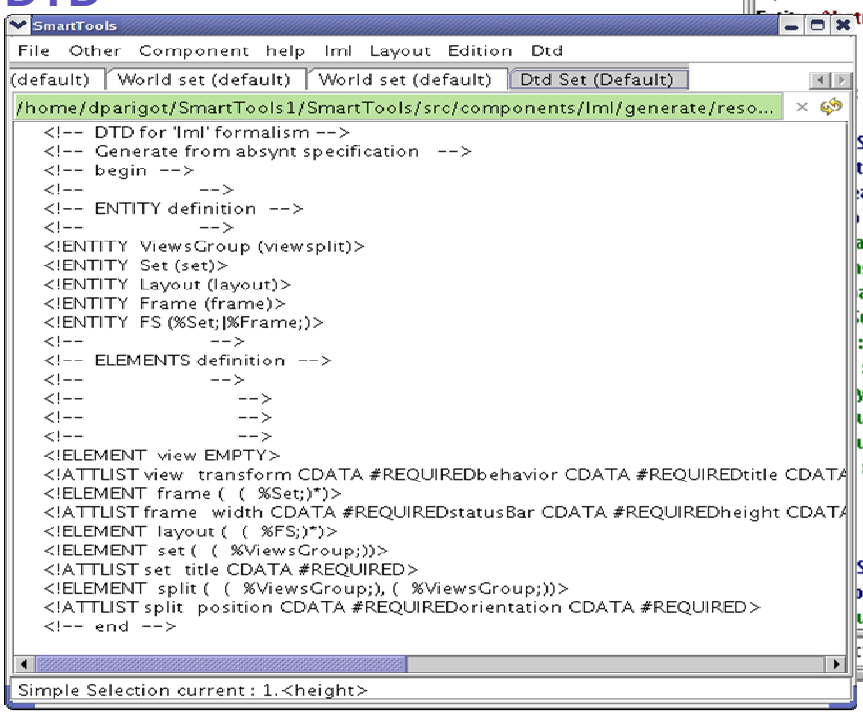
XML Schema



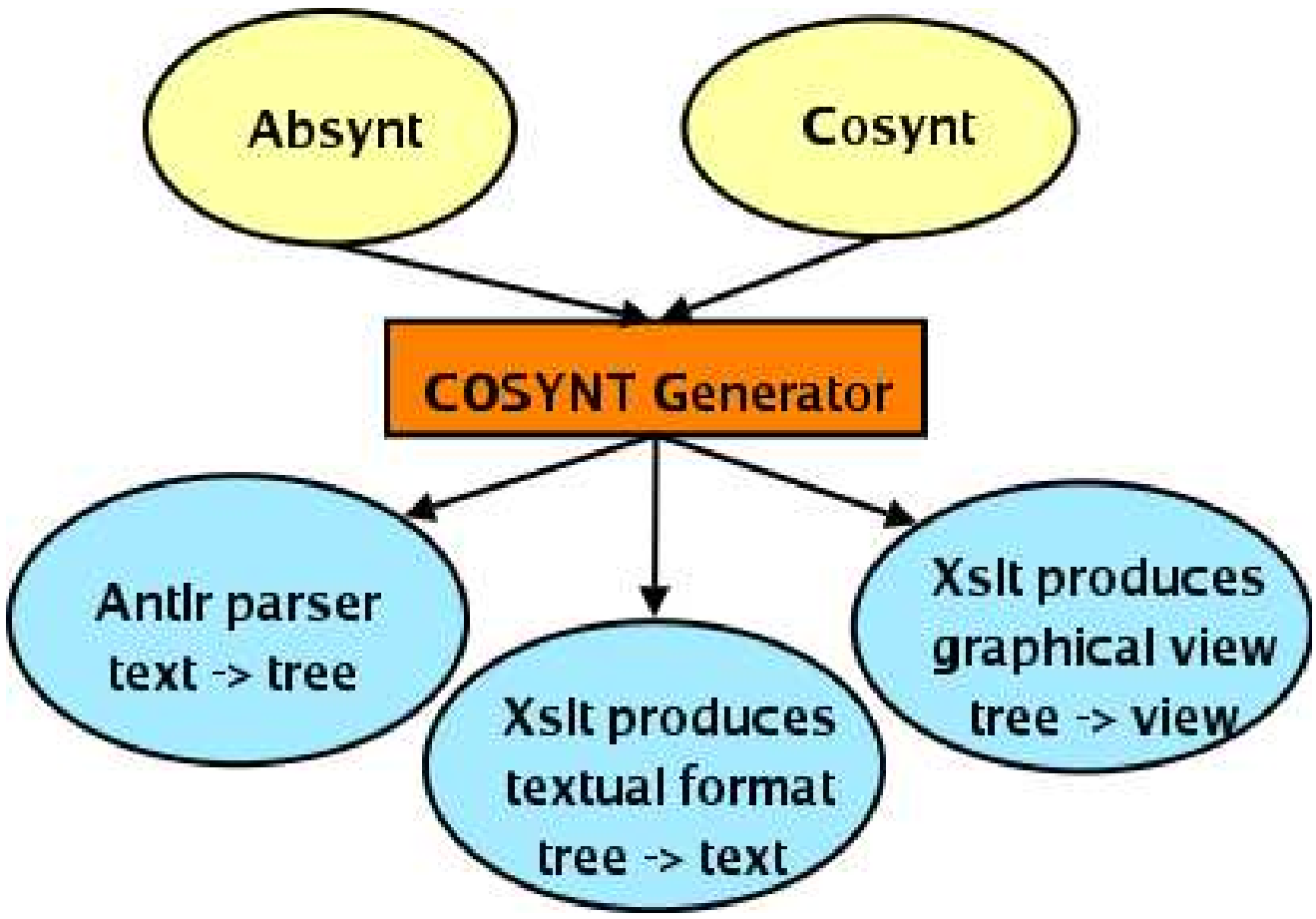
UML



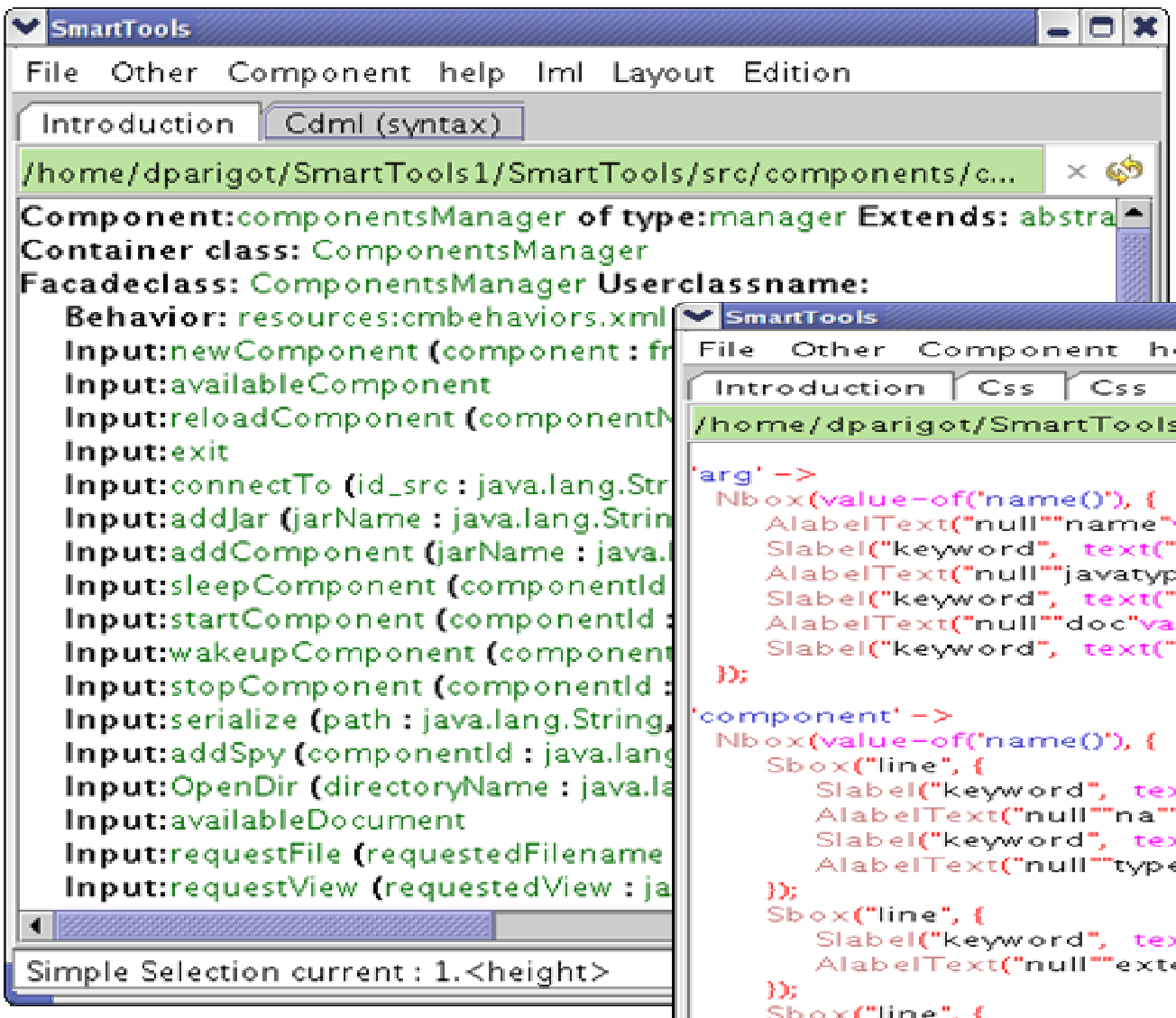
DTD



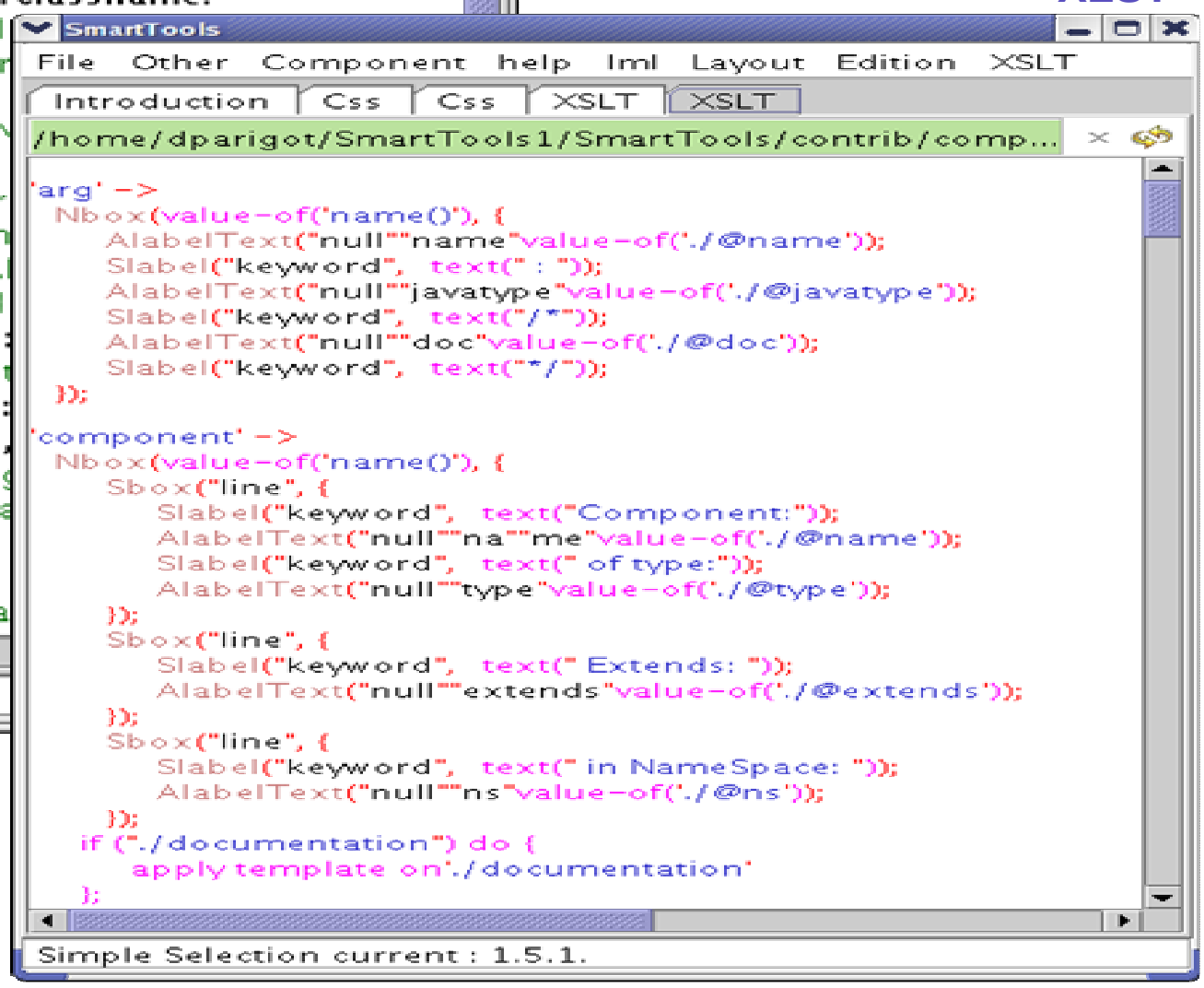
DSL [Cosynt] pour définir divers formats



Éditeur



XLST

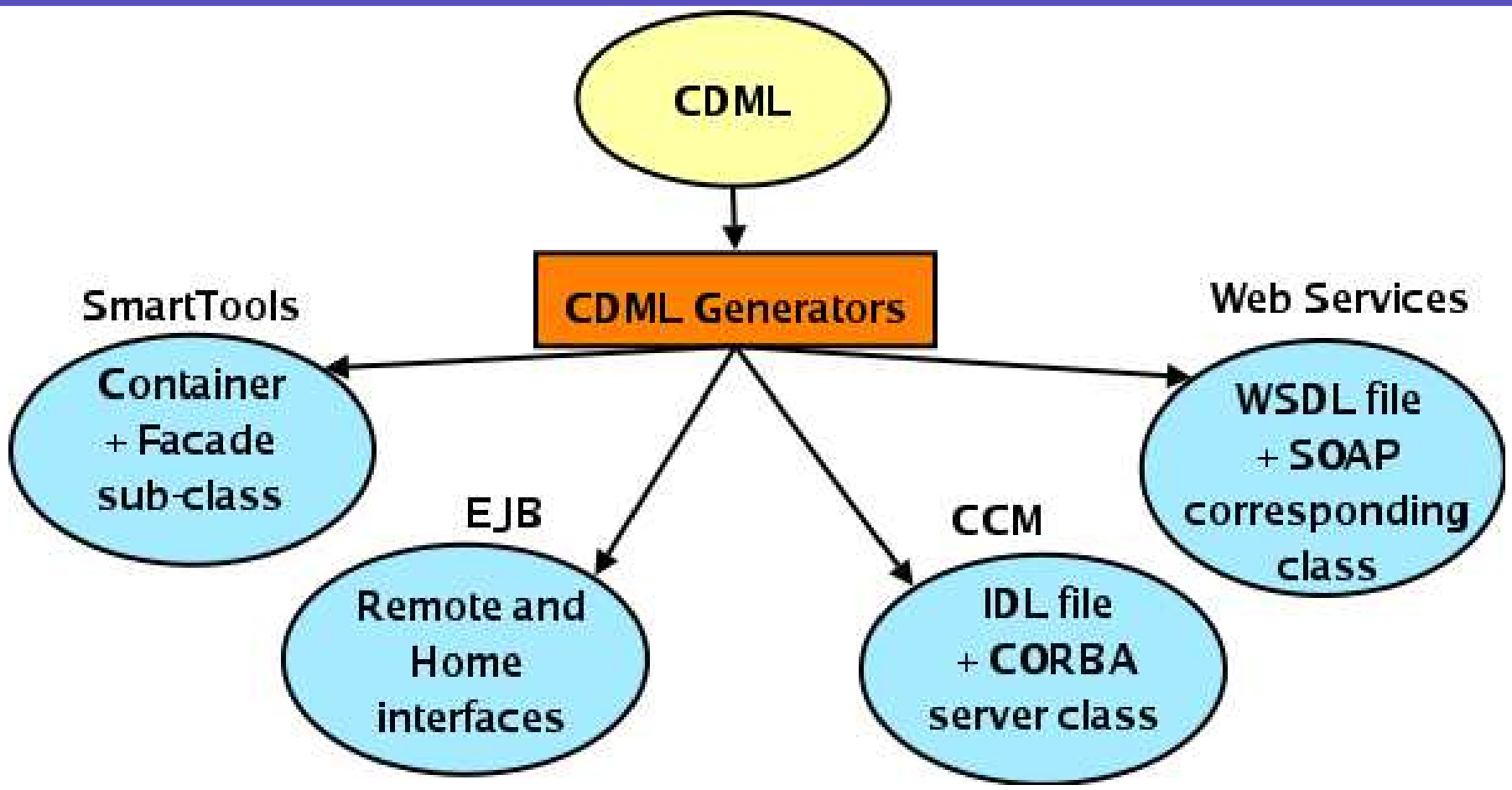


SmartTools

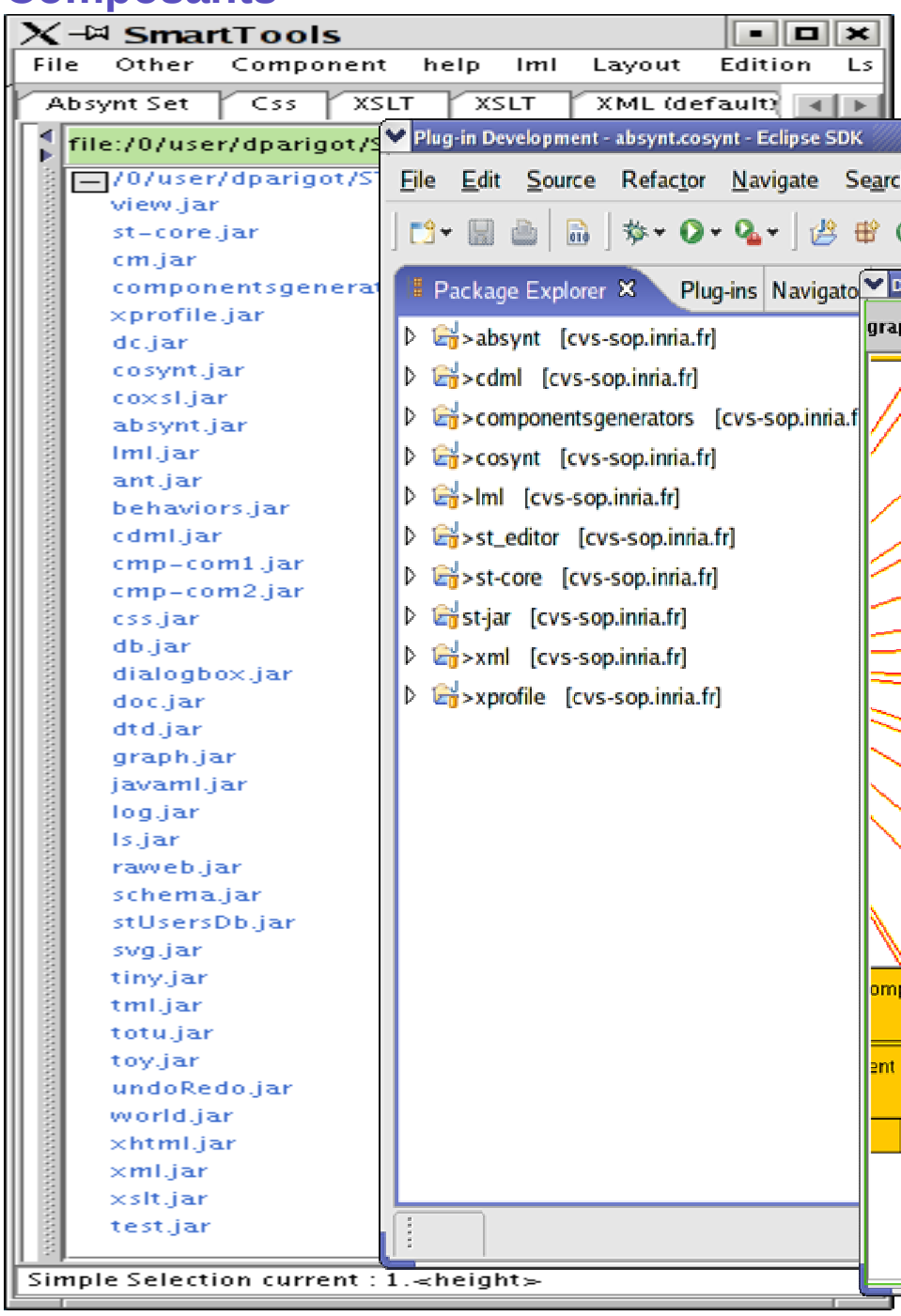
INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA
SOPHIA ANTIPOLIS

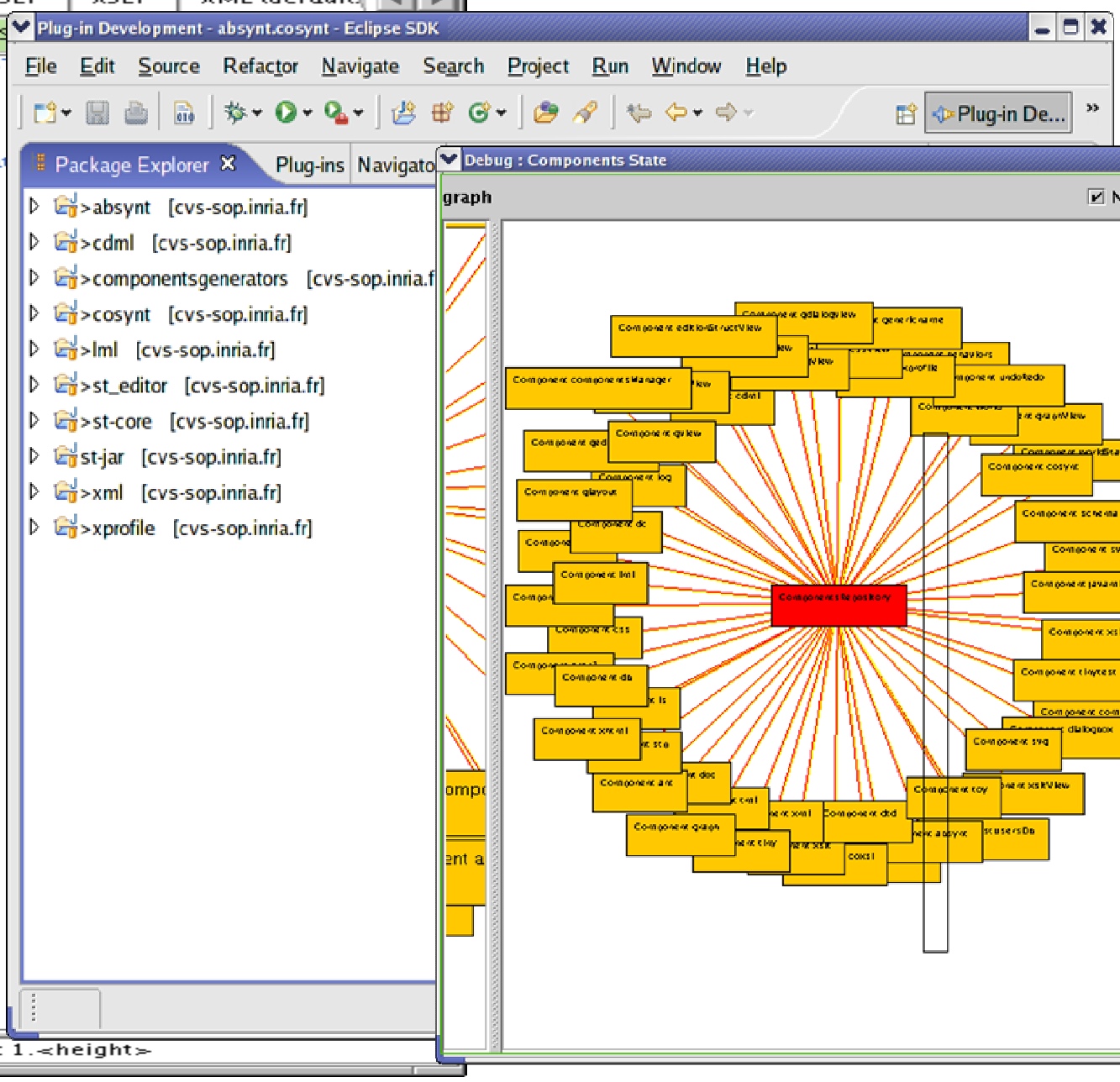
DSL [CDML] pour définir des composants



Composants



Plugins Eclipse



Type de composants

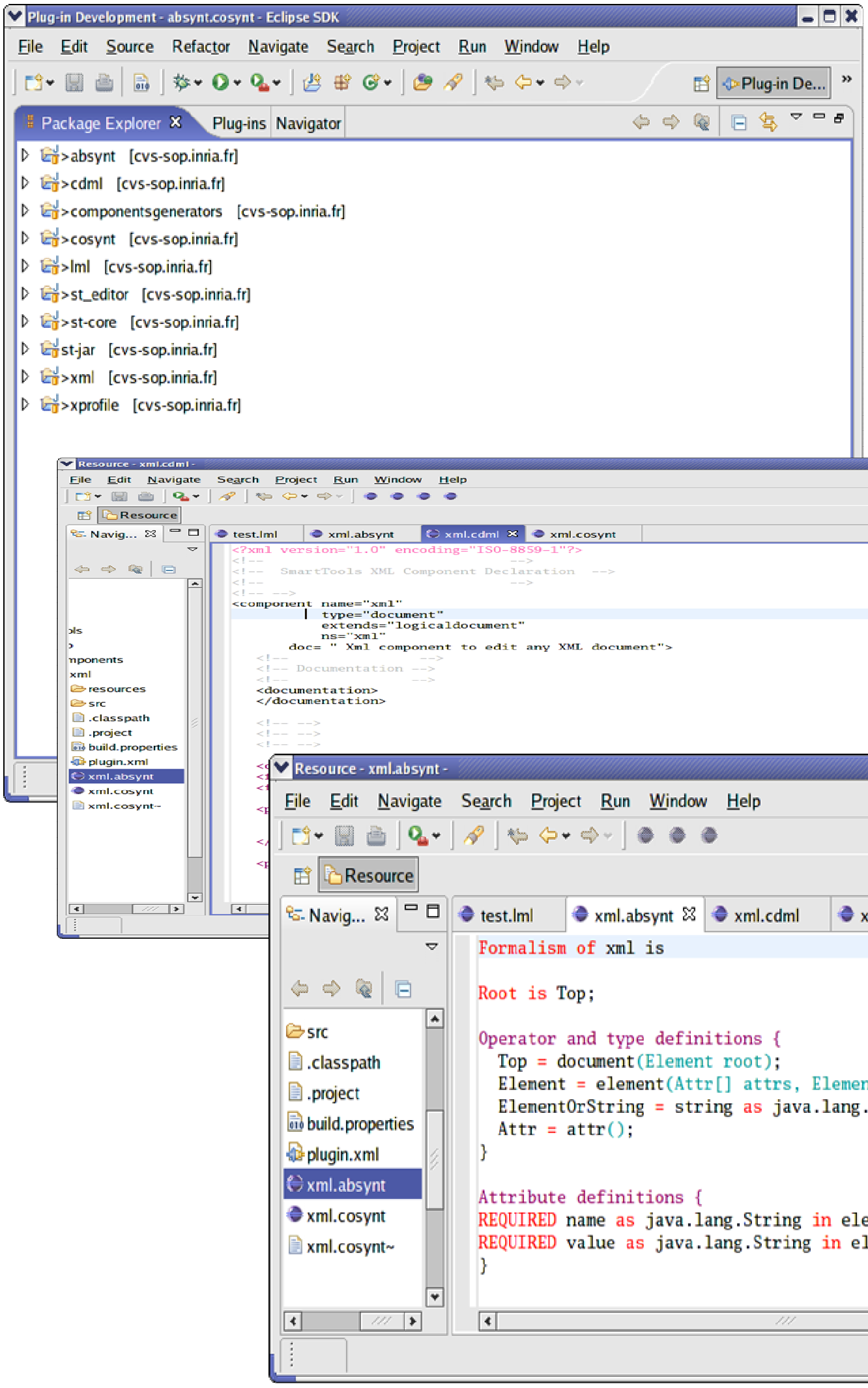
SmartTools

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA
SOPHIA ANTIPOLIS

SmartTools dans Eclipse

Résultat d'un mois de travail !



Exemples de syntaxe de surface [Cosynt]

Absynt de CDML

SmartTools

File Other Component help Iml Layout Edition Absynt

Introduction | Cdml (syntax) | Cdml (default) | Lml Set (default) | Absynt Set

/home/dparigot/SmartTools1/SmartTools/contrib/components/cdml/cdml.absynt

```
Formalism of cdml is
Root is Top;
 Operator and type definitions{
   Top=    component (Documentation ? doc, ContainerClass ? containerclass, Faca
   Formalism=    formalism ();
   ContainerClass=    containerclass ();
   FacadeClass=    facadeclass ();
   Parser=    parser (Extension [] extension);
   Extension=    extention ();
   Lml=    Iml ();
   Behavior=    behavior ();
   Dependance=    dependance ();
   Attribute=    attribute (),   arg (),   binding (Attribute [] parameter);
   Port=    input (Attribute [] parameter),   output (Attribute [] parameter),   inout (A
   Documentation=    documentation as java.lang.String;
 }
 Attribute definitions{
   REQUIRED name as java.lang.String in formalism, containerclass, facadeclass, exte
   REQUIRED classname as java.lang.String in parser;
   REQUIRED type as java.lang.String in parser, component;
   REQUIRED extends as java.lang.String in component;
   REQUIRED file as java.lang.String in Iml, behavior;
```

Simple Selection current : 1.<height>

Cosynt de CDML

SmartTools

File Other Component help Iml Layout Edition Cosynt

Introduction | Schema (default) | Schema (default) | Cdml (default) | Cdml (syntax) | Cosynt Set (default)

/0/user/dparigot/ST5/SmartTools/contrib/components/cdml/cdml.cosynt

```
Cosynt for cdml is
  Concrete Syntaxe {
    BNF {
      component(Doc,Containerclass,Facadeclass,Formalism,Parser,Lml,Behavior,Depe
      dependance [@jar,@name] : '' @jar @name;
      documentation : %VAR;
      formalism [@name] : '' @name;
      parser(Extension) [@type,@generator,@classname] : *[ #Extension ] @type @gener
      extention [@name] : '' @name;
      facadeclass [@name,@userclassname] : '' @name @userclassname;
      attribute [@javatype,@name,@doc] : @name ' : ' @javatype;
      containerclass [@name] : '' @name;
      arg [@javatype,@name] : @name ':toto' @javatype;
      behavior [@file] : '' @file;
      lml [@file,@name] : '' @file @name;
      input(Parameter) [@method] : @method *[ #Parameter separator ' , ' beforeList '('
      inout(Parameter) [@outputname,@name,@outputmethod,@method] : *[ #Paramete
      binding(Parameter) [@toMethod] : *[ #Parameter ] @toMethod;
      output(Parameter) [@method] : @method *[ #Parameter separator ' , ' beforeList '(
    }
    Parser
      [k=1]{
      }
    Lexer
      [k=1,attributes=VAR]{
        VAR=;
      }
  }
  Layout {
```

Simple Selection current : 1.6.1.<position>

XSLT produit

```
'arg' ->
  Nbox(value-of('name()'), {
      AlabelText("null""name"value-of('./@name'));
      Slabel("keyword",  text(" : "));
      AlabelText("null""javatype"value-of('./@javatype'));
      Slabel("keyword",  text("/*"));
      AlabelText("null""doc"value-of('./@doc'));
      Slabel("keyword",  text("*/"));
  });

'component' ->
  Nbox(value-of('name()'), {
      Sbox("line", {
          Slabel("keyword",  text("Component:"));
          AlabelText("null""na""me"value-of('./@name'));
          Slabel("keyword",  text(" of type:"));
          AlabelText("null""type"value-of('./@type'));
      });
      Sbox("line", {
          Slabel("keyword",  text(" Extends: "));
          AlabelText("null""extends"value-of('./@ex
      });
      Sbox("line", {
          Slabel("keyword",  text(" in NameSpace: "
          AlabelText("null""ns"value-of('./@ns'));
      });
    if ("./documentation") do {
       apply template on"./documentation"
    };
```

Simple Selection current : 1.5.1.

Exemple

SmartTools

File Other Component help Iml Layout Edition Cdml

Introduction | Cdml (syntax) | Cdml (syntax)

/home/dparigot/SmartTools1/SmartTools/src/core/fr/smarttools/core/view/resources/glayout.cdml

Component:glayout **of type:**view **Extends:** abstractView **in NameSpace:** fr.smarttools.core.view
Container class: GLayoutContainer
Facadeclass: GdocLayout **Userclassname:**
  Input:statusInfo (statusText : java.lang.String)
  Input:statusWarn (statusText : java.lang.String)
  Output:modify (modif : java.lang.String/**/, syntaxe : java.lang.String/**/, attributes : java.util.HashMap/**/)

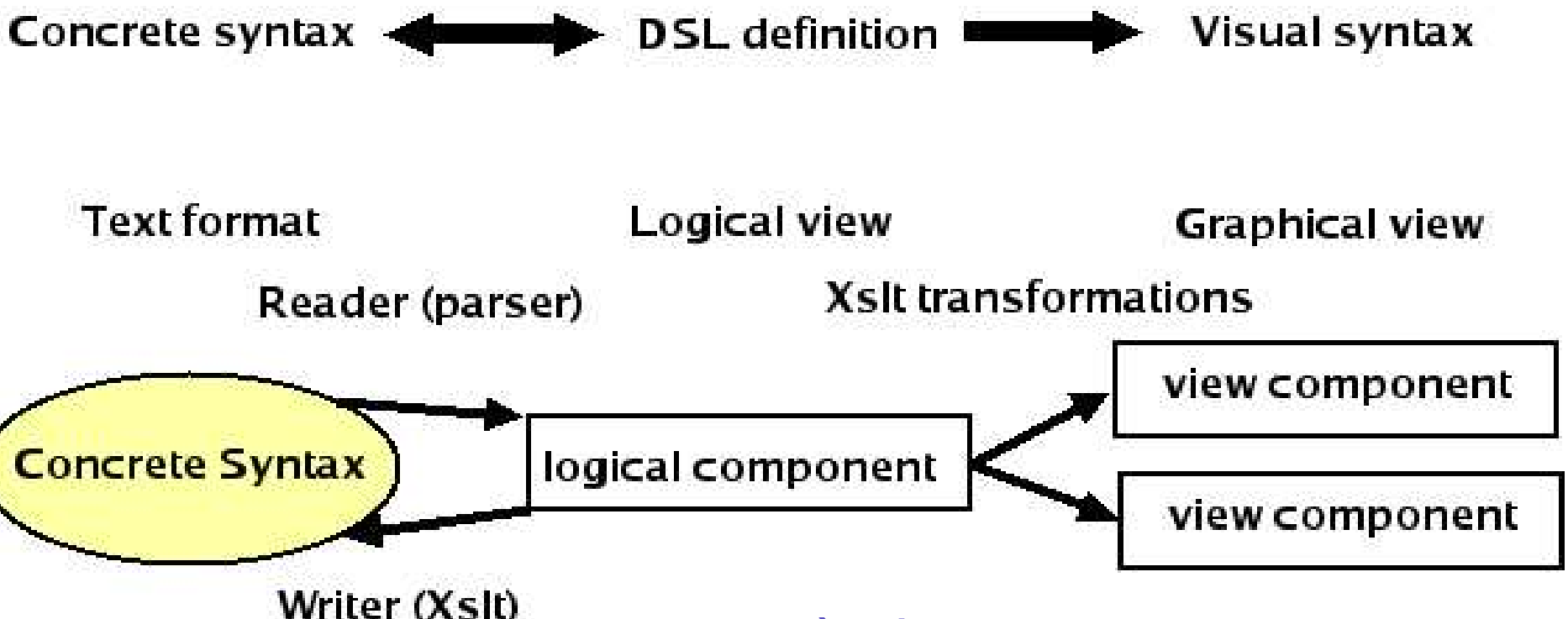
Simple Selection current : 1.<height>

SmartTools

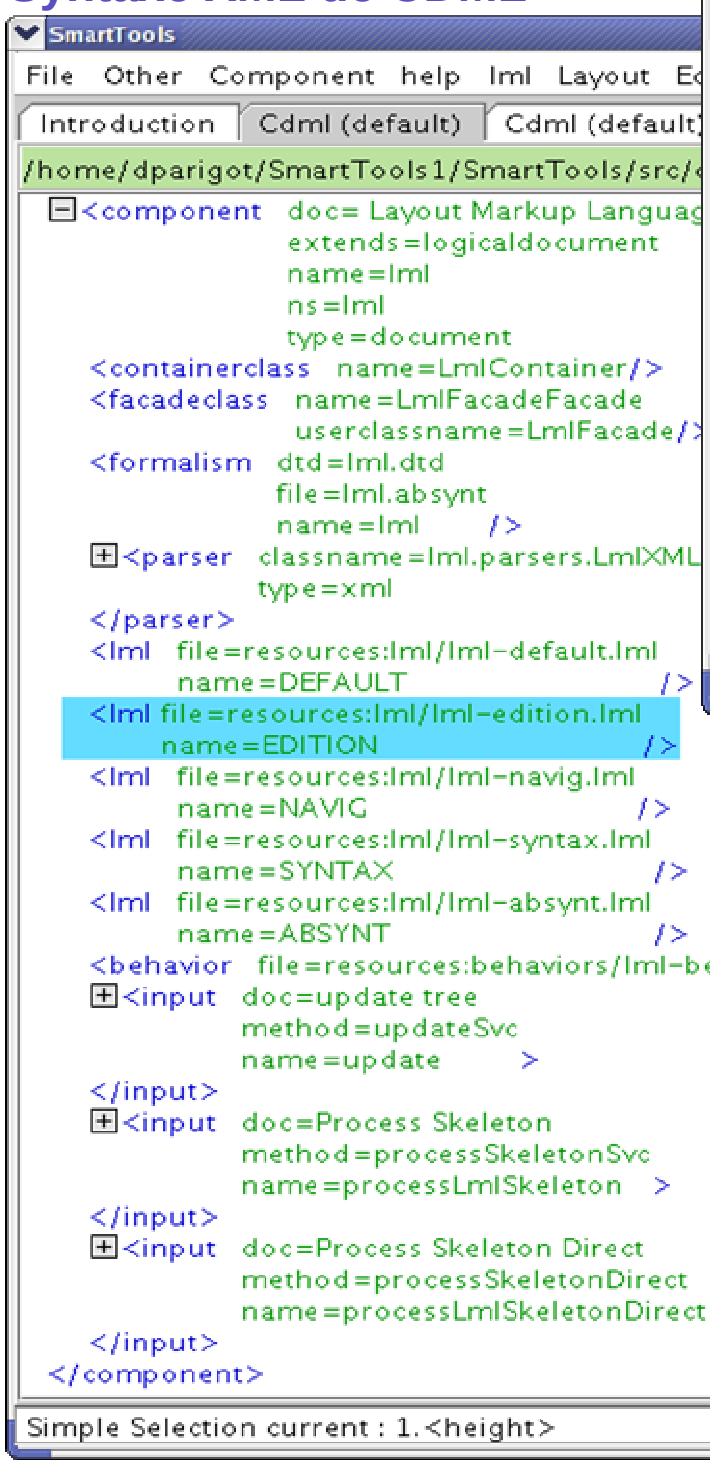
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA SOPHIA ANTIPOLIS

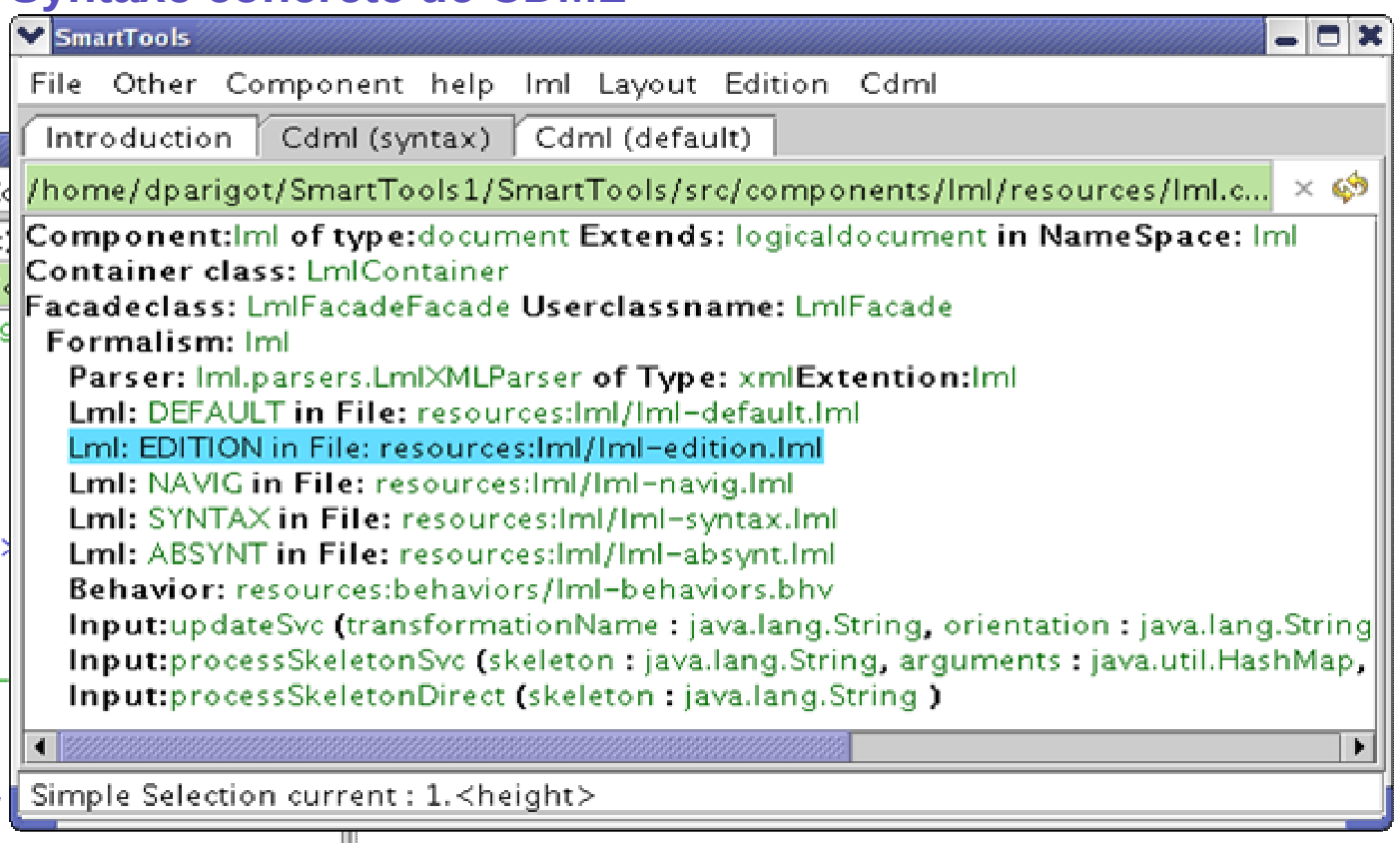
DSL [Cosynt] pour définir diverses représentations



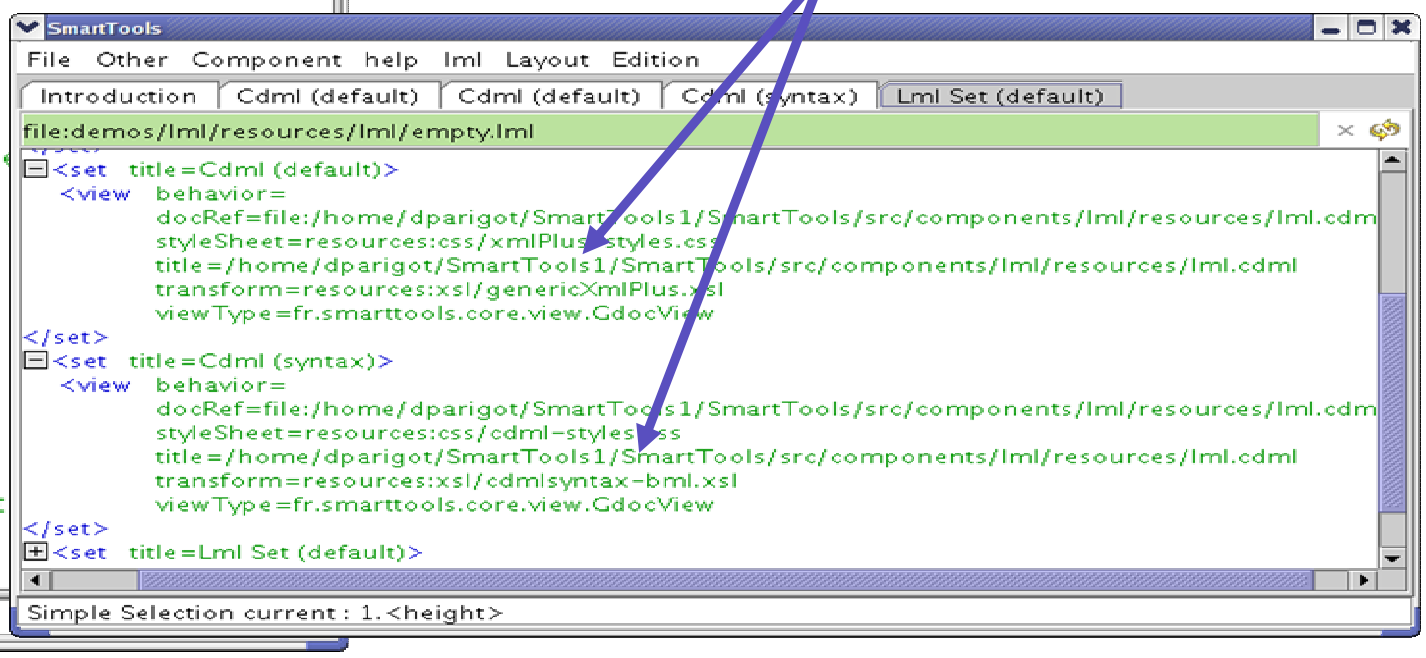
Syntaxe XML de CDML



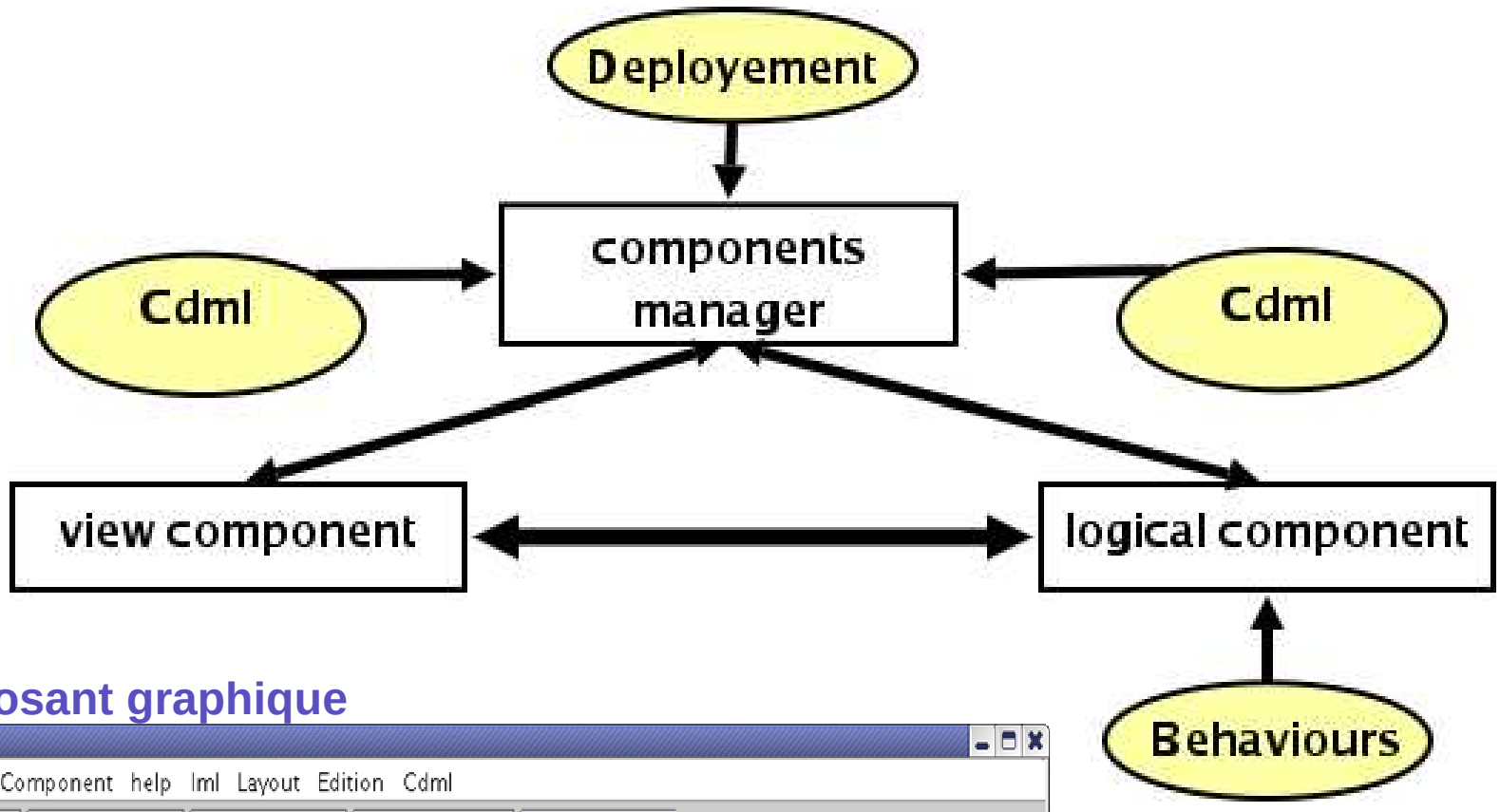
Syntaxe concrète de CDML



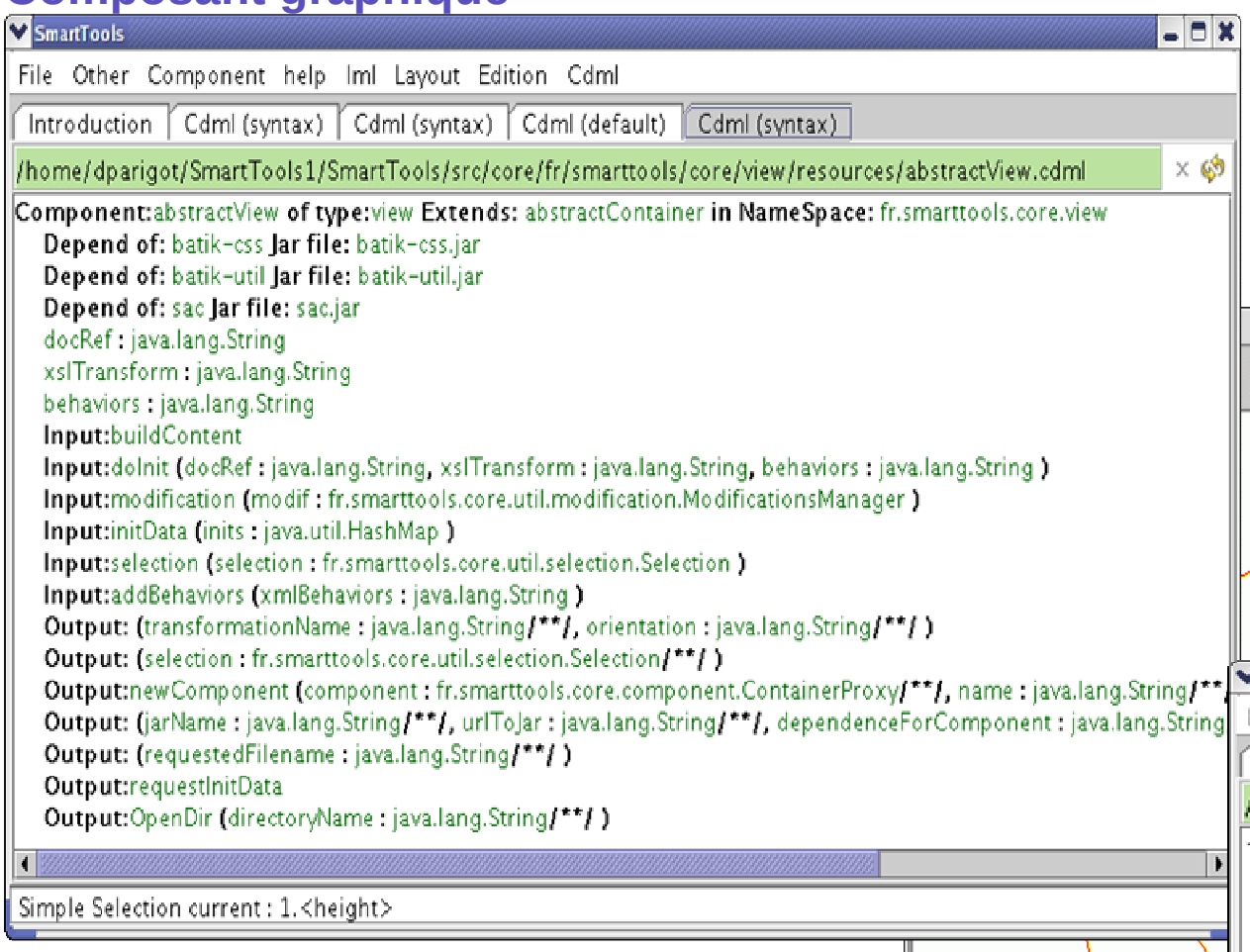
Transformations XSLT



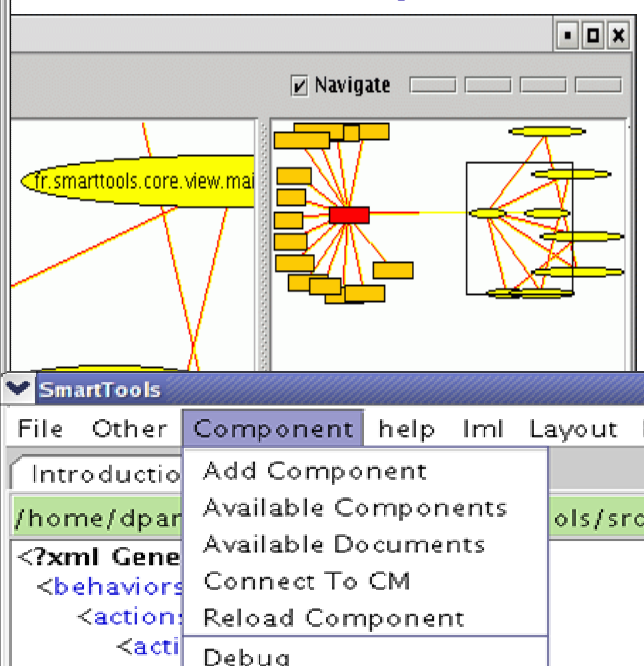
Architecture dirigée par les services



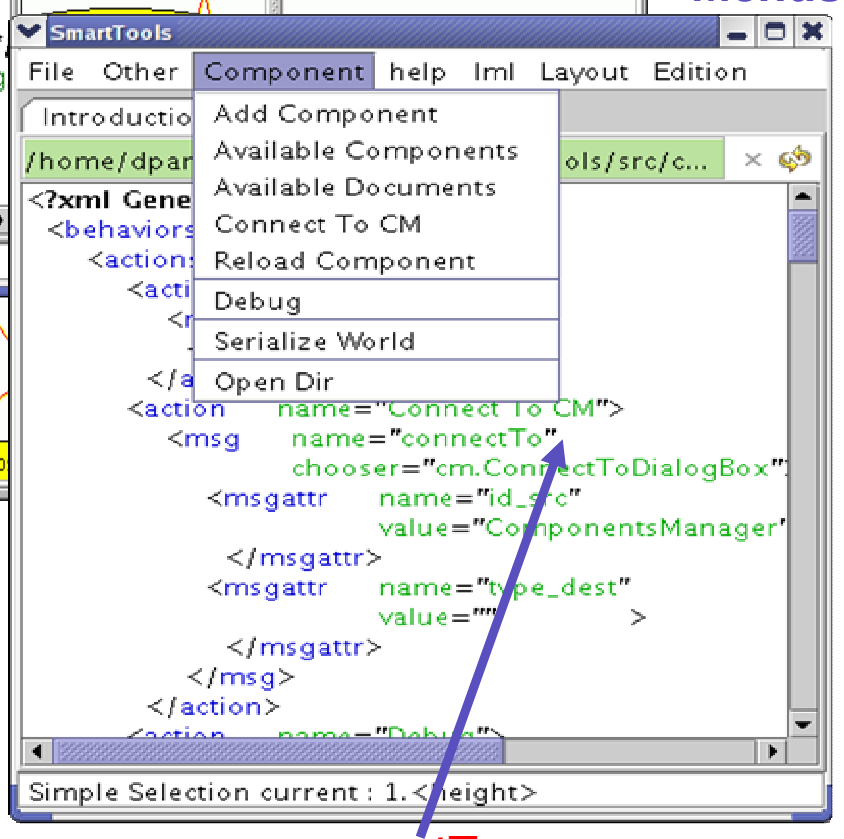
Composant graphique



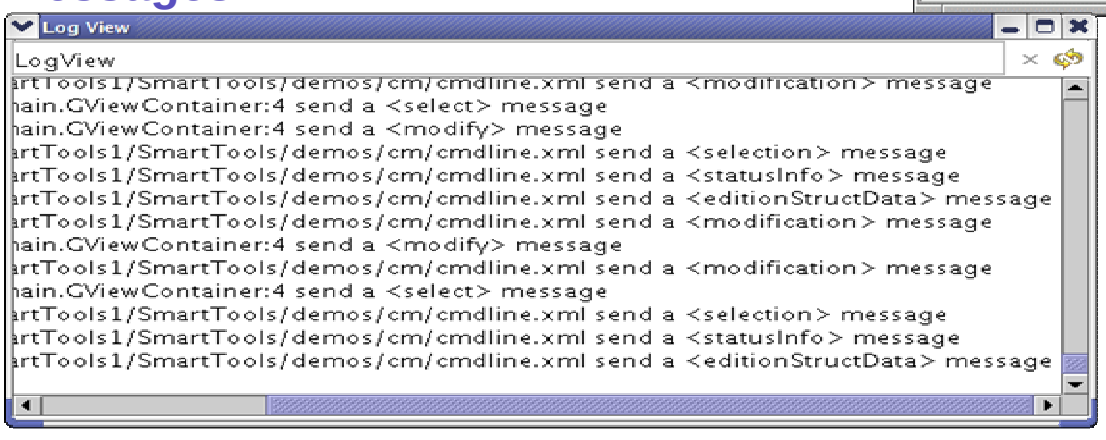
Graphes des composants



Menus

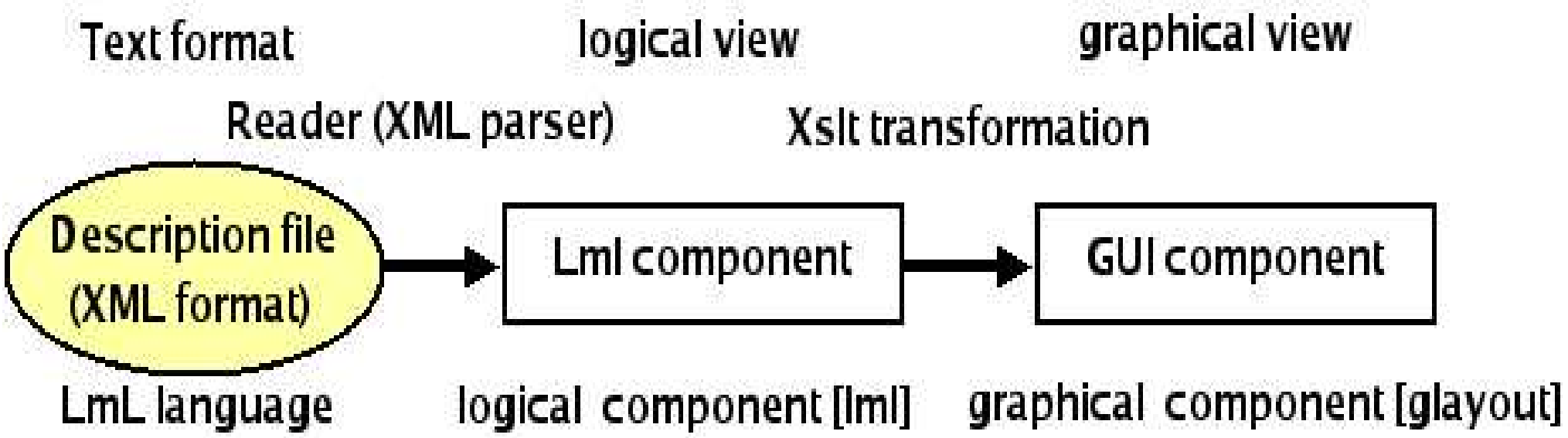


Messages

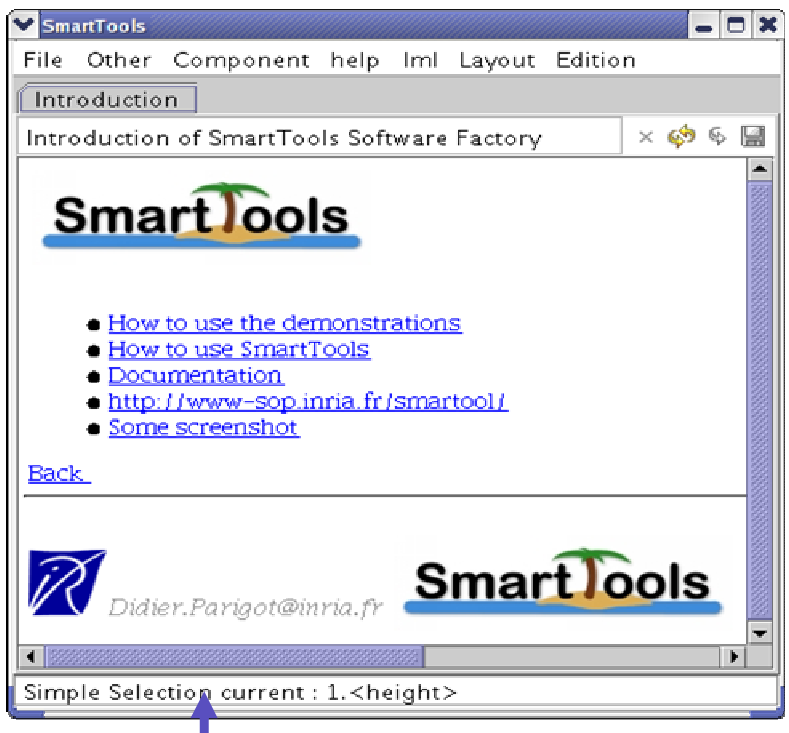


connectTo

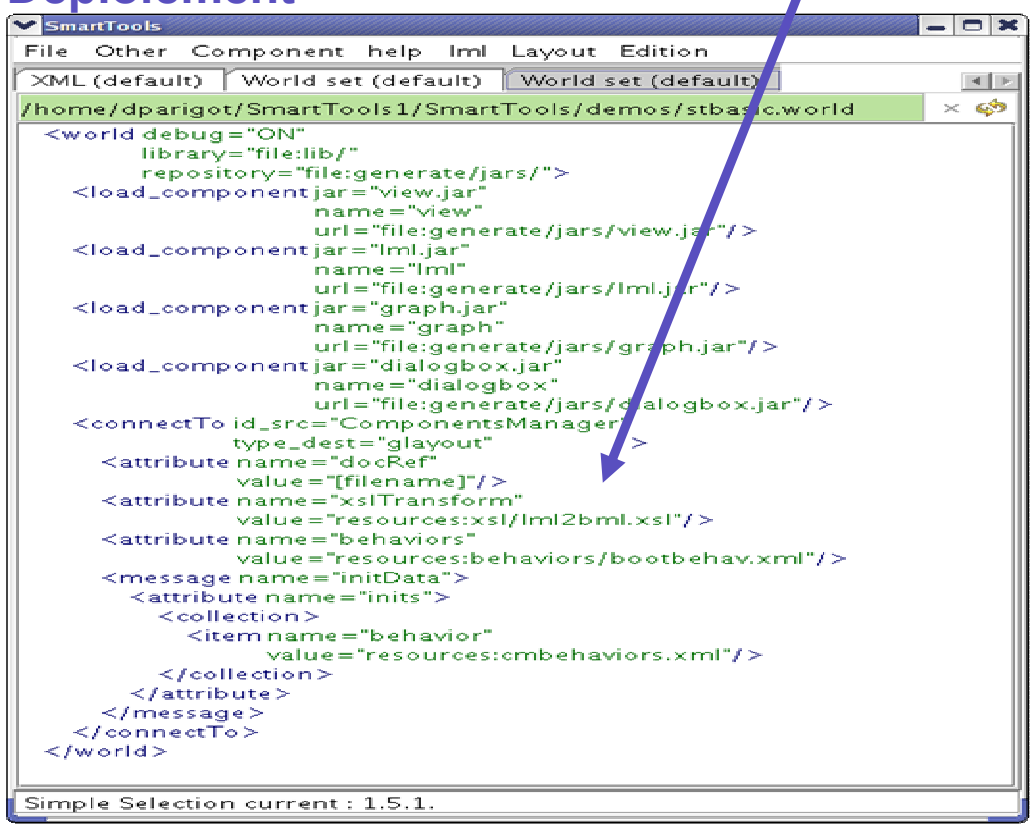
DSL [LML] pour définir une GUI



HTML

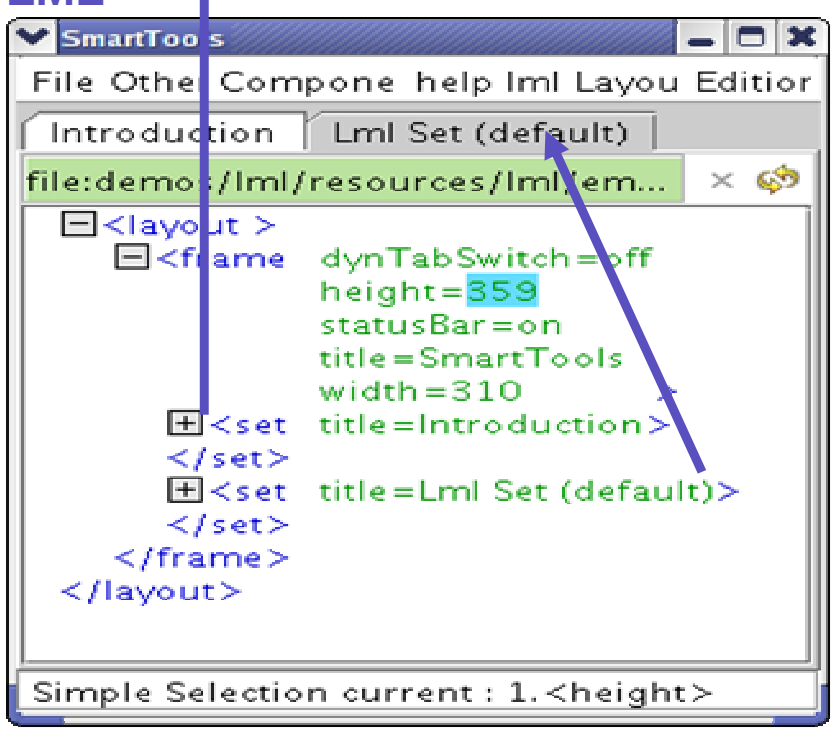


Déploiement

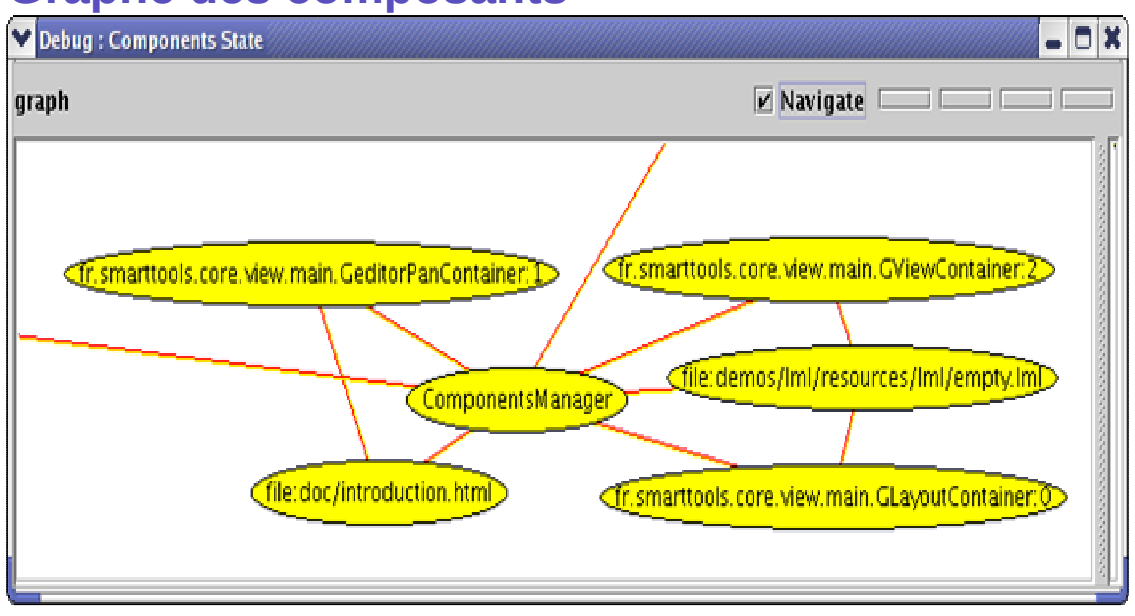


Transformation XSLT

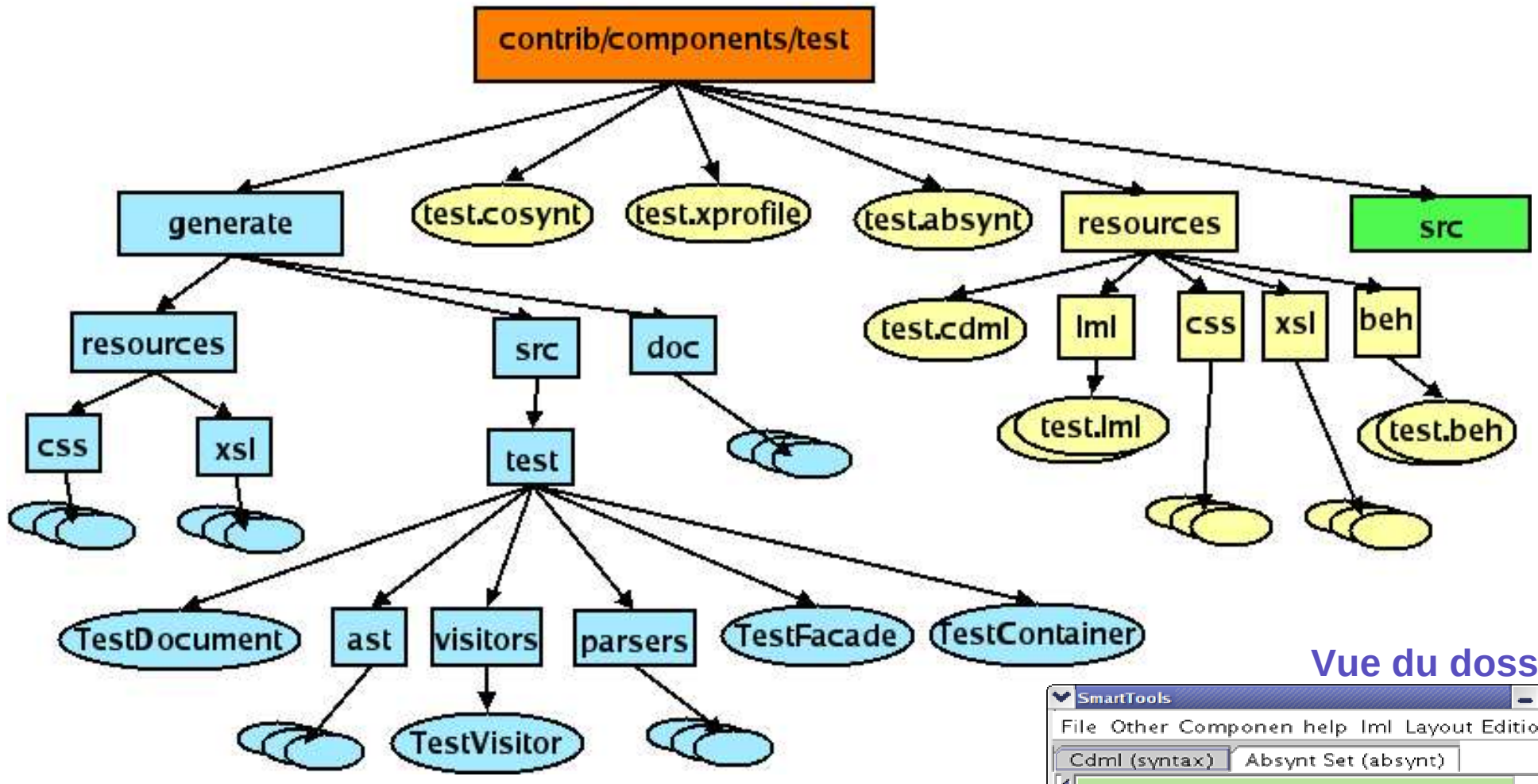
LML



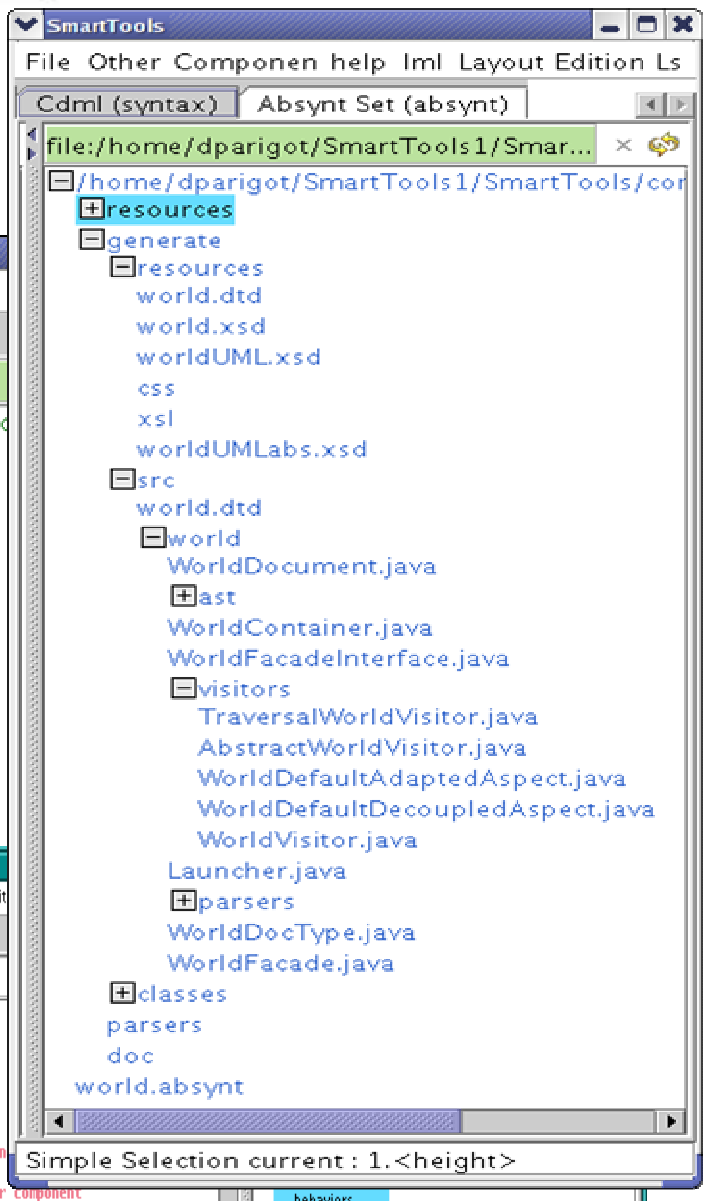
Graphe des composants



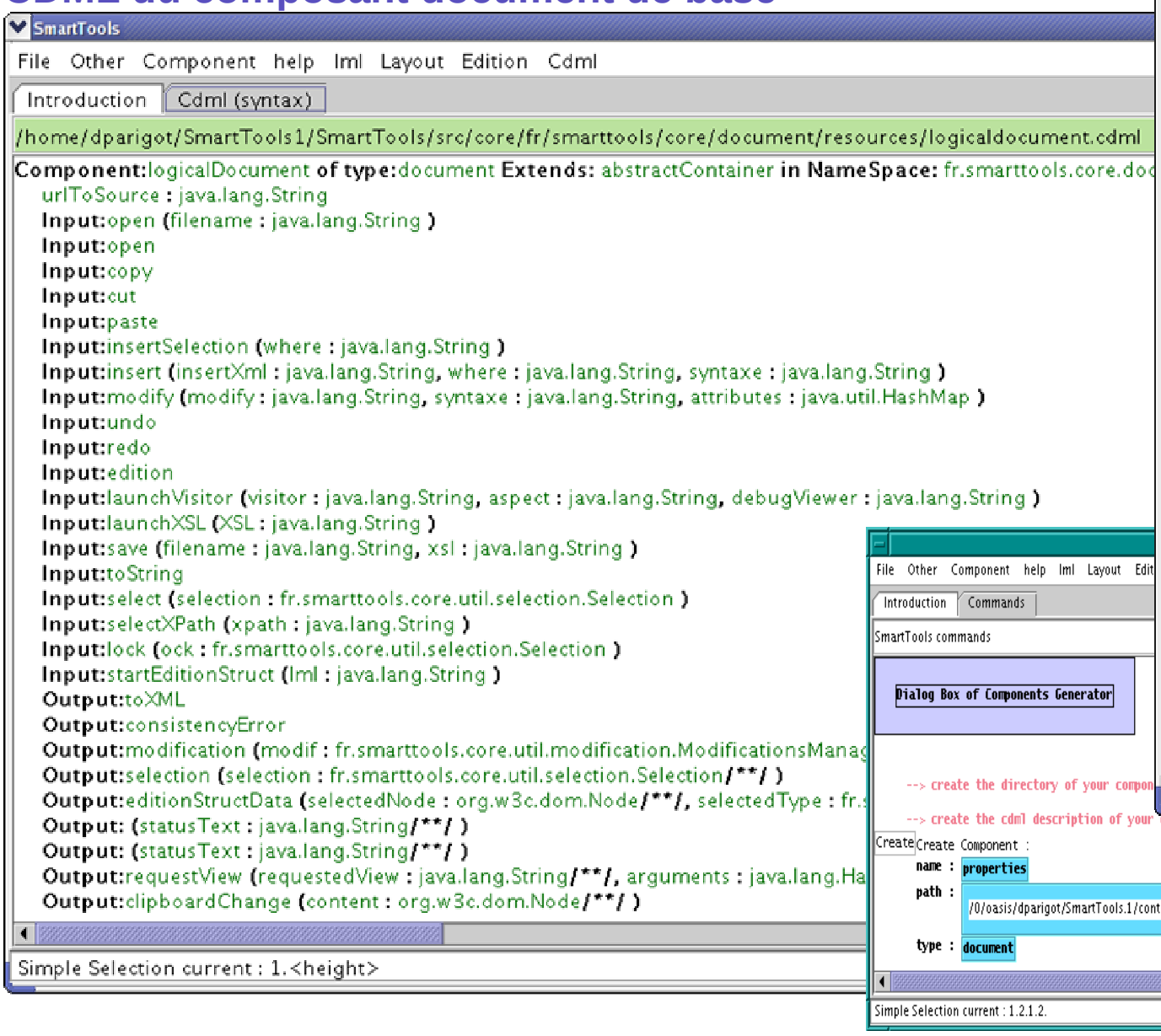
Génération de 90% du code des applications



Vue du dossier

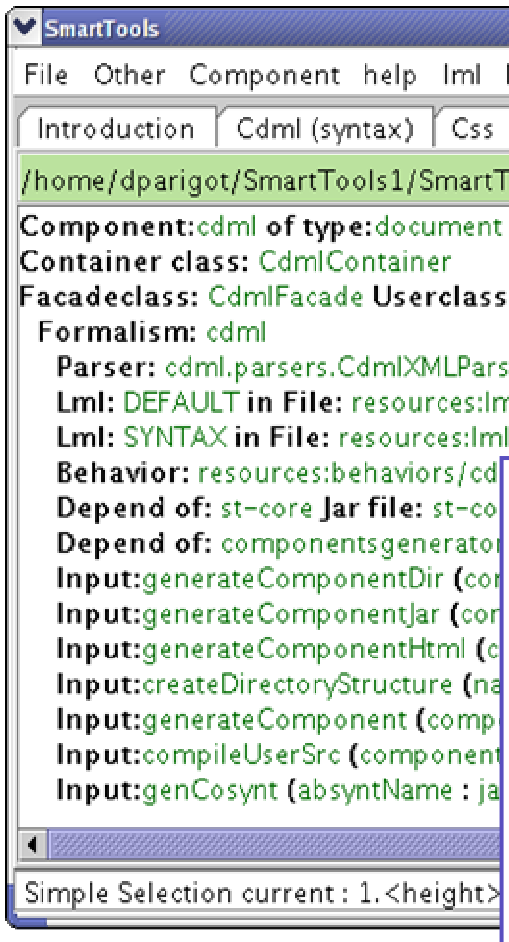


CDML du composant document de base

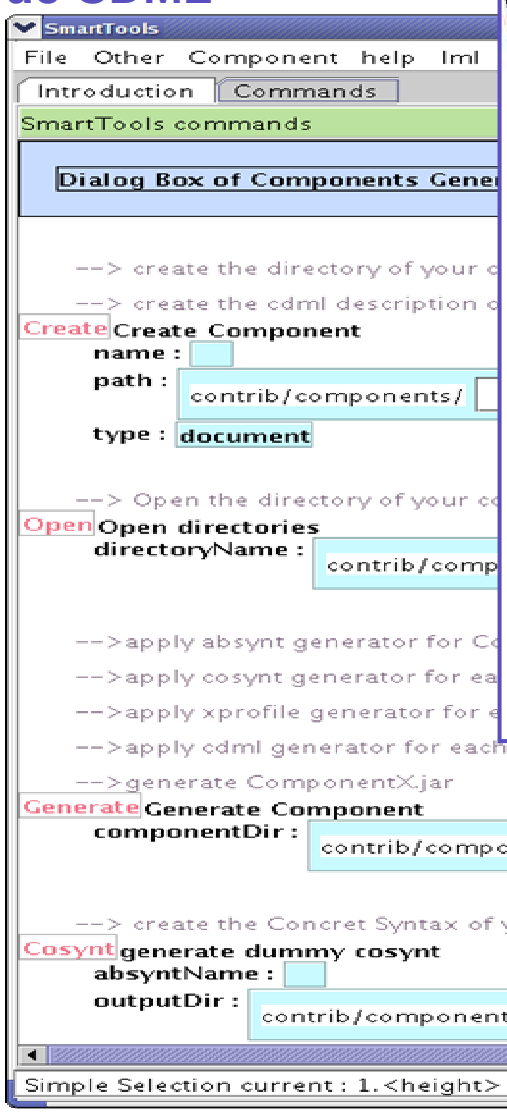


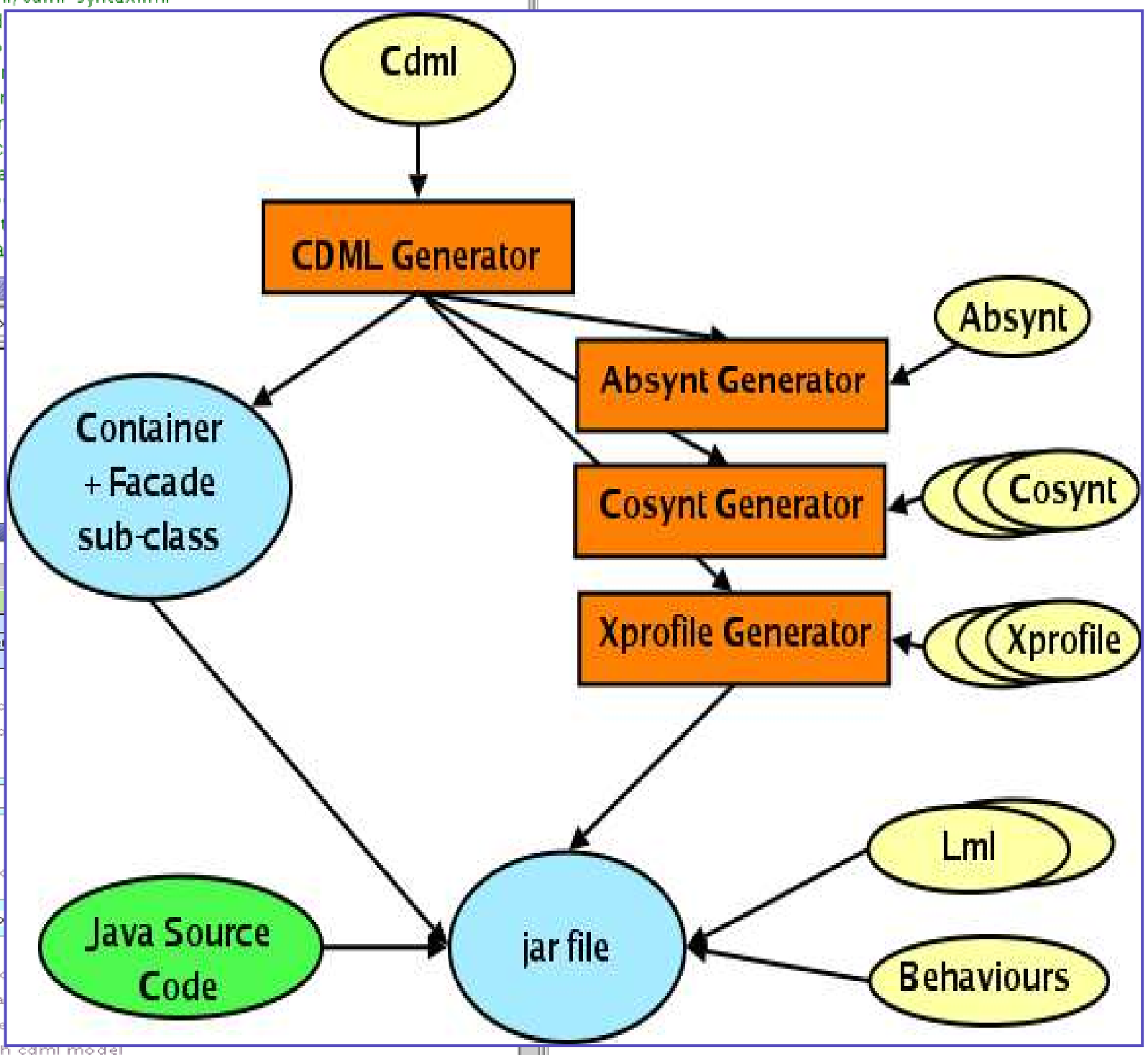
Développement dirigé par les DSLs

CDML de CDML

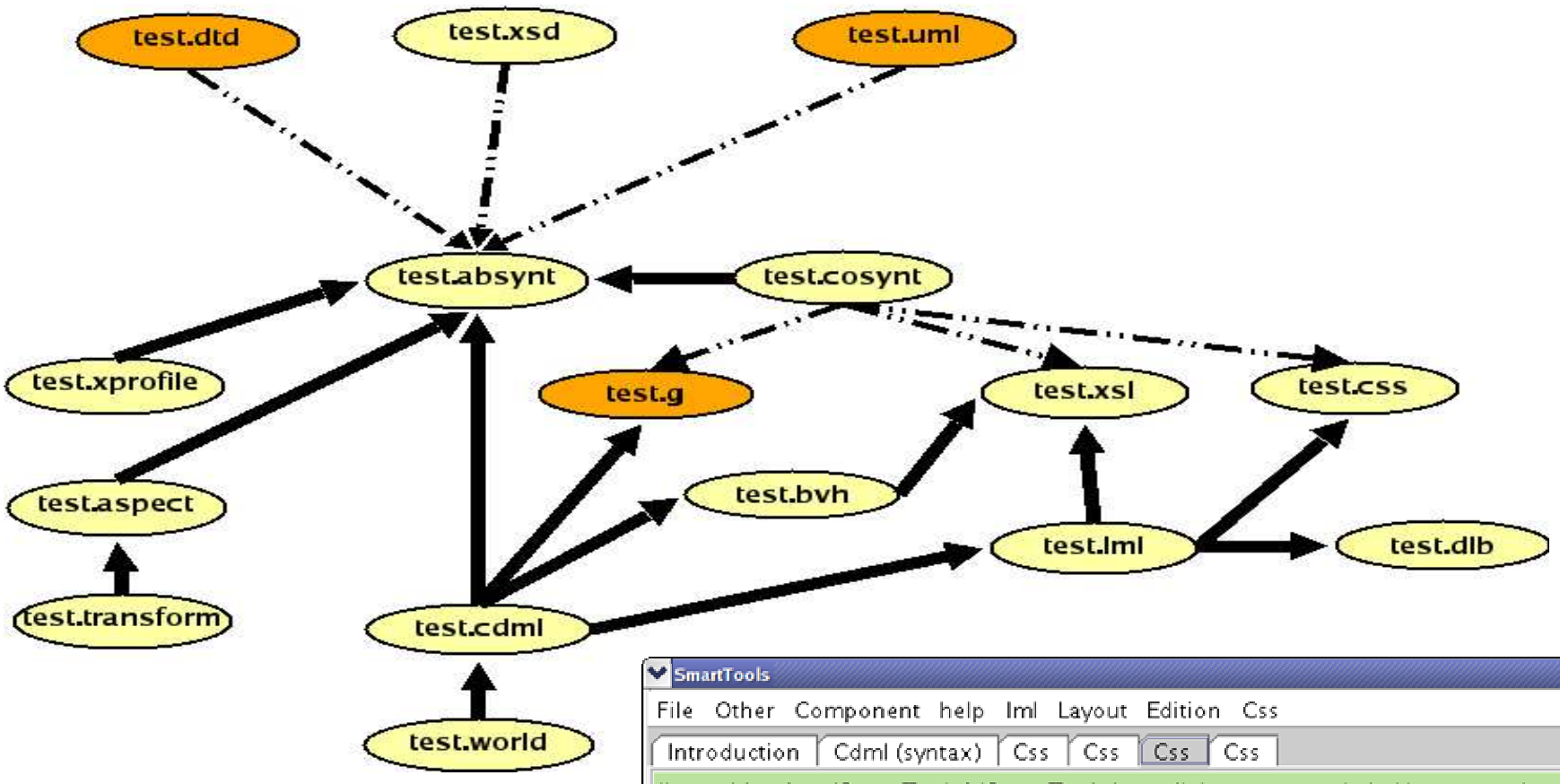


Boite de dialogue de CDML



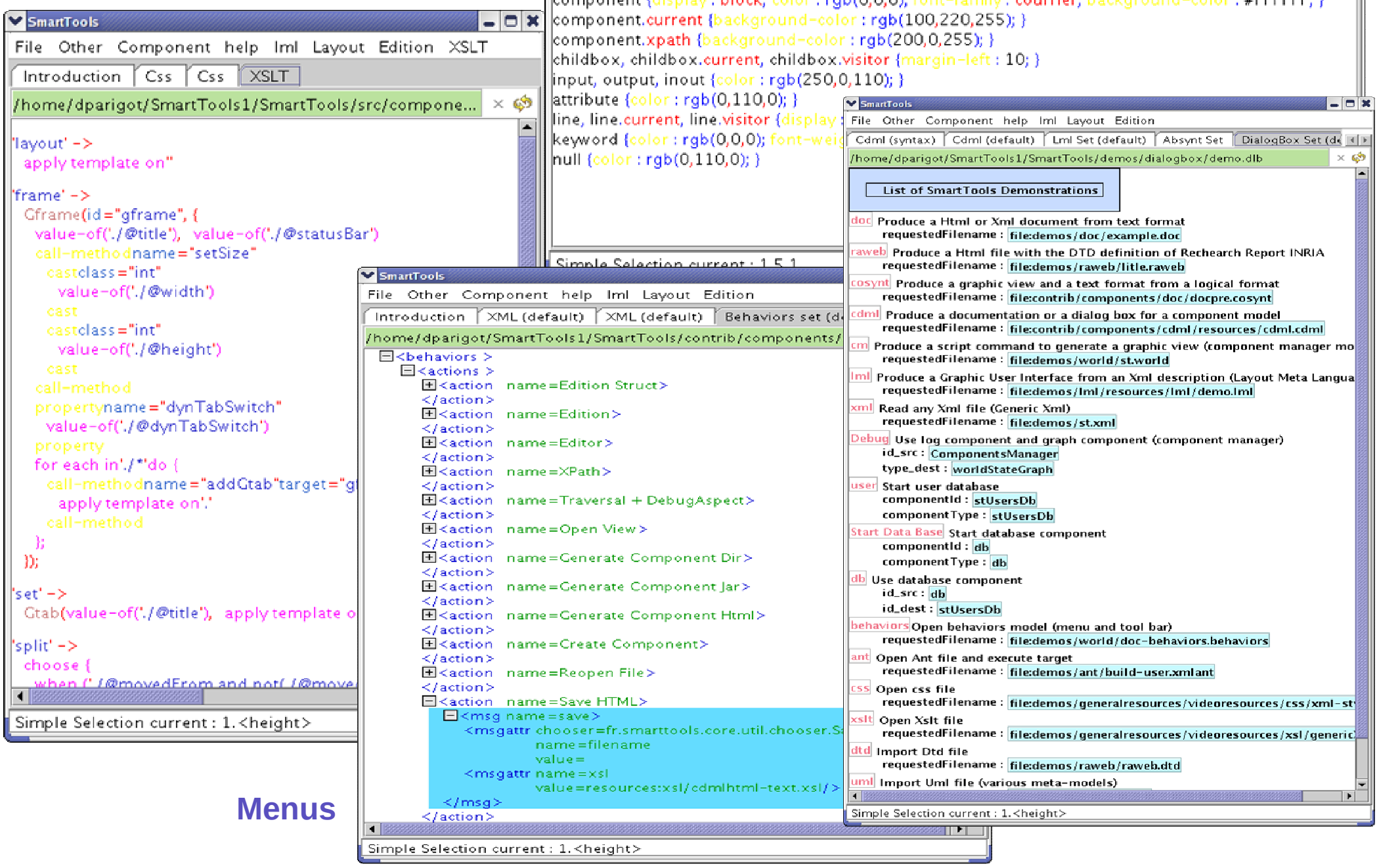


Schémas de dépendance entre les DSLs



CSS

XSLT



Menus

Les fortes évolutions de l'informatique

Les applications d'hier

programmes

langages

compilateurs

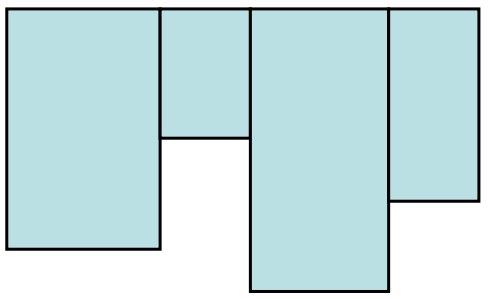
exécution

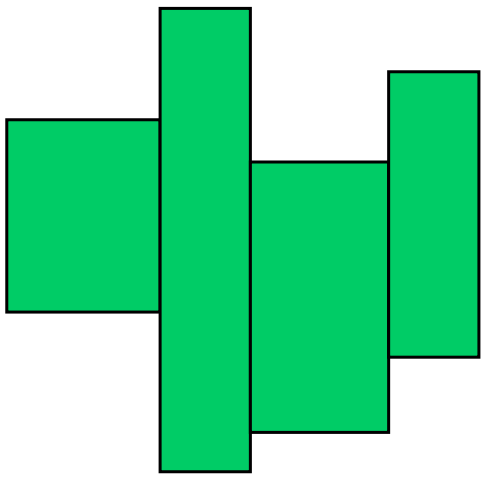
système

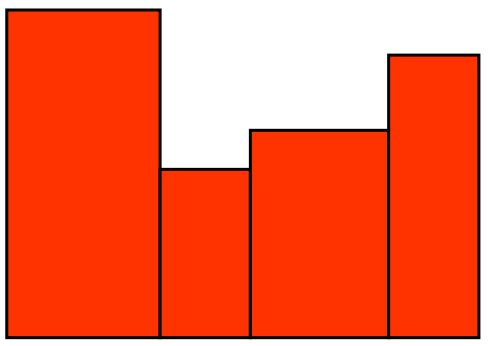
Technologies logicielles

Technologies matérielles

Les applications de demain



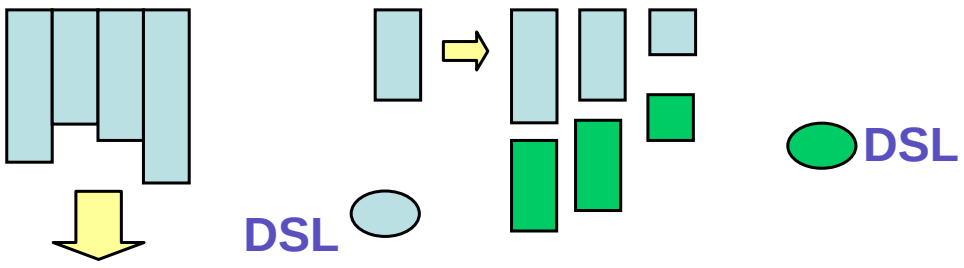




Les langages de programmation ont évolués beaucoup moins vite

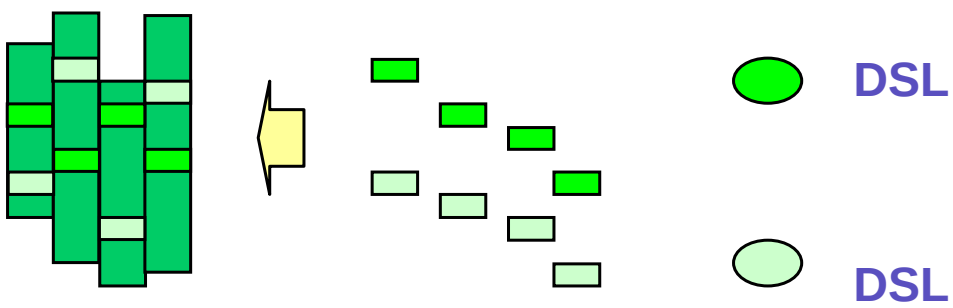
Les quatre concepts

Programmation dirigée par des DSLs



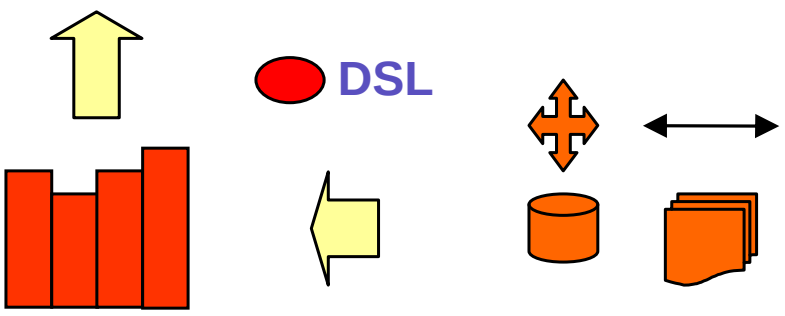
MDA
Ingénierie Dirigée
par les Modèles

Programmation par préoccupations



AOP
Programmation
par aspects

Programmation par composants



SOA
Architecture Orientée
Services

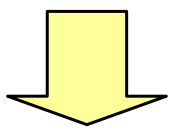
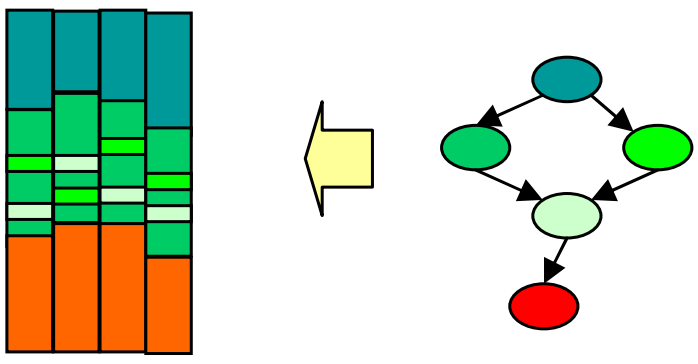


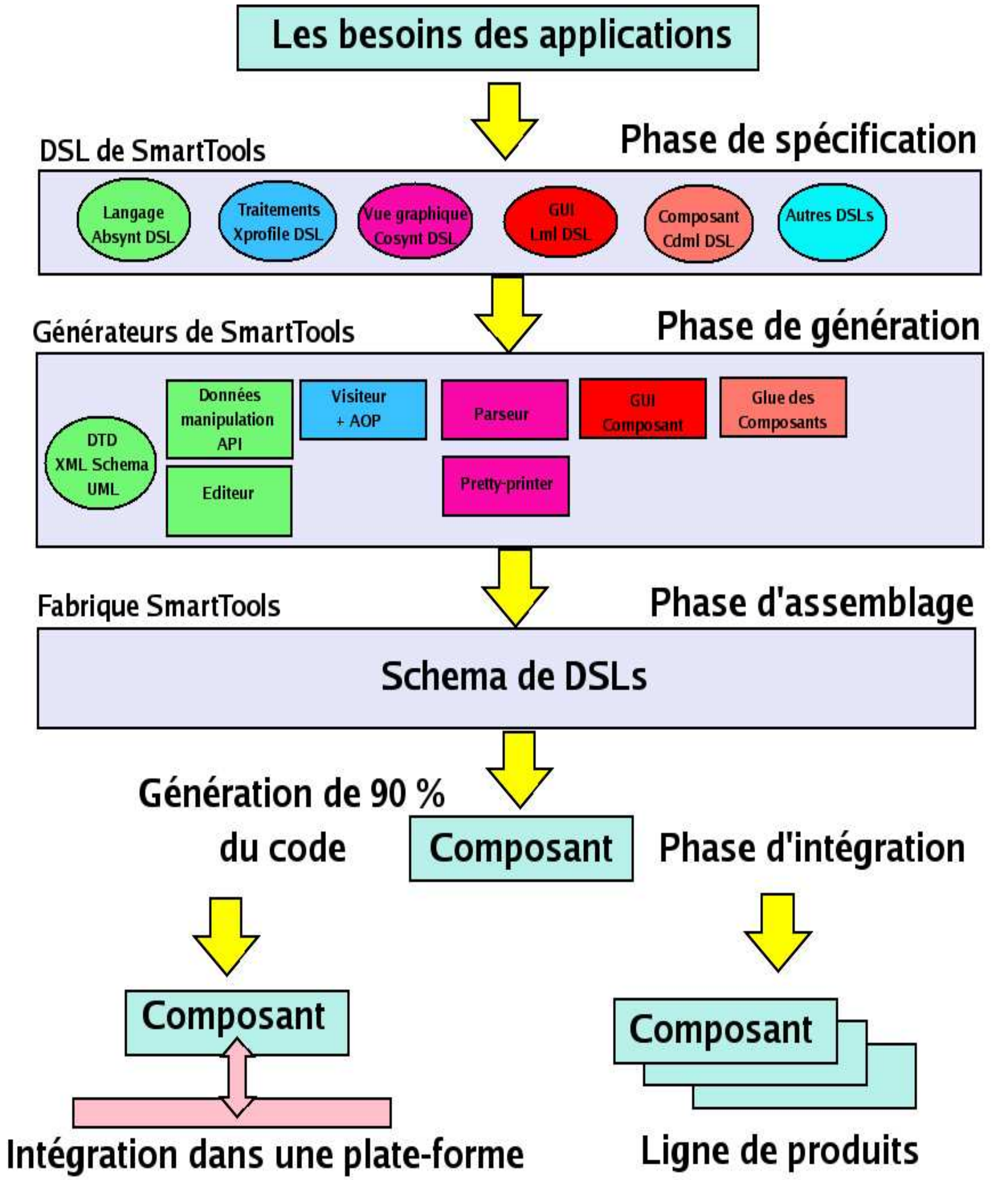
Schéma d'assemblage des DSLs



DSLs

FABRIQUE LOGICIELLE

Développement en quatre phases



La programmation de demain sera :

Tout par composants

Tout par assemblage dynamique

Tout par langages, abstraction

- DSL métiers

Tout par transformations

- Transformation de modèles

Tout par analogies

- Projection des traitements

Tout par séparation des préoccupations

- Une programmation pointilliste

Une programmation incrémentale

- Spécification – Implémentation

Une macro programmation

- Systèmes complexes, évolutifs, multipartenaires

Les programmes ne seront plus manipulées directement par leurs interfaces (API)

Les facteurs de ce bouleversement

Fortes évolutions :

- Technologies logicielles
- Technologies matérielles
- Communauté d'utilisateurs

Nouveaux besoins et exigences :

- Automatiser le développement
- Réduire le temps du développement
- Améliorer la qualité du logiciel
- Augmenter le retour d'investissement
- Intégrer rapidement les nouvelles technologies

Principaux obstacles :

- Évolution des langages de programmation
- L'existant des systèmes informatiques
- La formation

Enjeux de ce bouleversement

Nécessité d'agir vite

- Mouvement international grâce aux Consortiums
- Marché en plein expansion (e-commerce)

Les enjeux économiques

- Les grands de l'informatique vont nous imposer leurs standards et outils

Nature des engagements

- De la matière grise (du logiciel)
- Un retour d'investissement rapide possible
- Solution envisageable/raisonnable à très court terme !