INRIA
SOPHIA ANTIPOLIS

Comment va-t-on programmer demain ?

Vite et Bien !

Macro Programmation

Fabrique Logicielle : SmartTools

Didier Parigot

Didier.Parigot@inria.fr

26 avril 2005

Contexte: Motivations

- **Rupture** dans le processus du développement logiciel

- Comprendre et **anticiper**
  - Analyser les problèmes
  - Proposer des solutions

- Passage d'une **micro** programmation (algorithmique) à une **macro** programmation (système complexe)

- **Automatisation** accrue du processus de développement
  - Fabrique logicielle

Contexte : Fortes évolutions

Les applications d'hier

Les applications de demain

programmes

langages

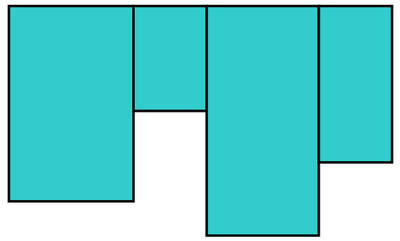
compilateurs

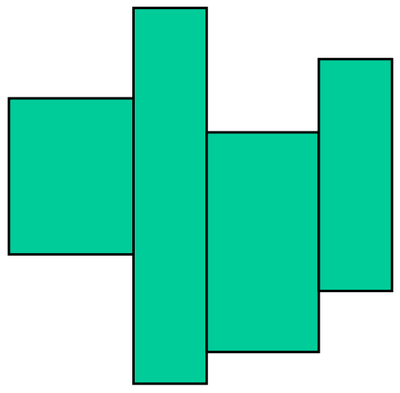
exécution

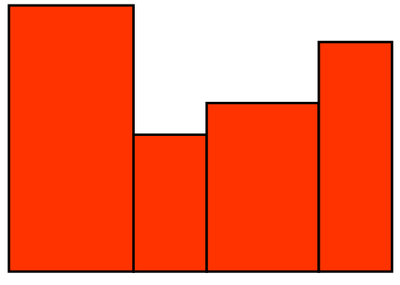
système

Technologies logicielles

Technologies matérielles

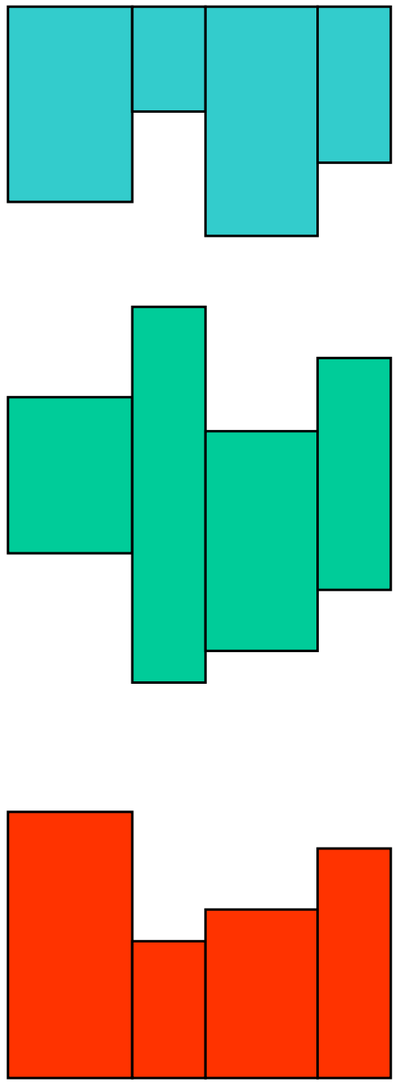






Rupture dans le développement

Les applications doivent être fortement évolutives



Patrons de conception

Par séparation de préoccupations

La programmation par aspects

La programmation par composants

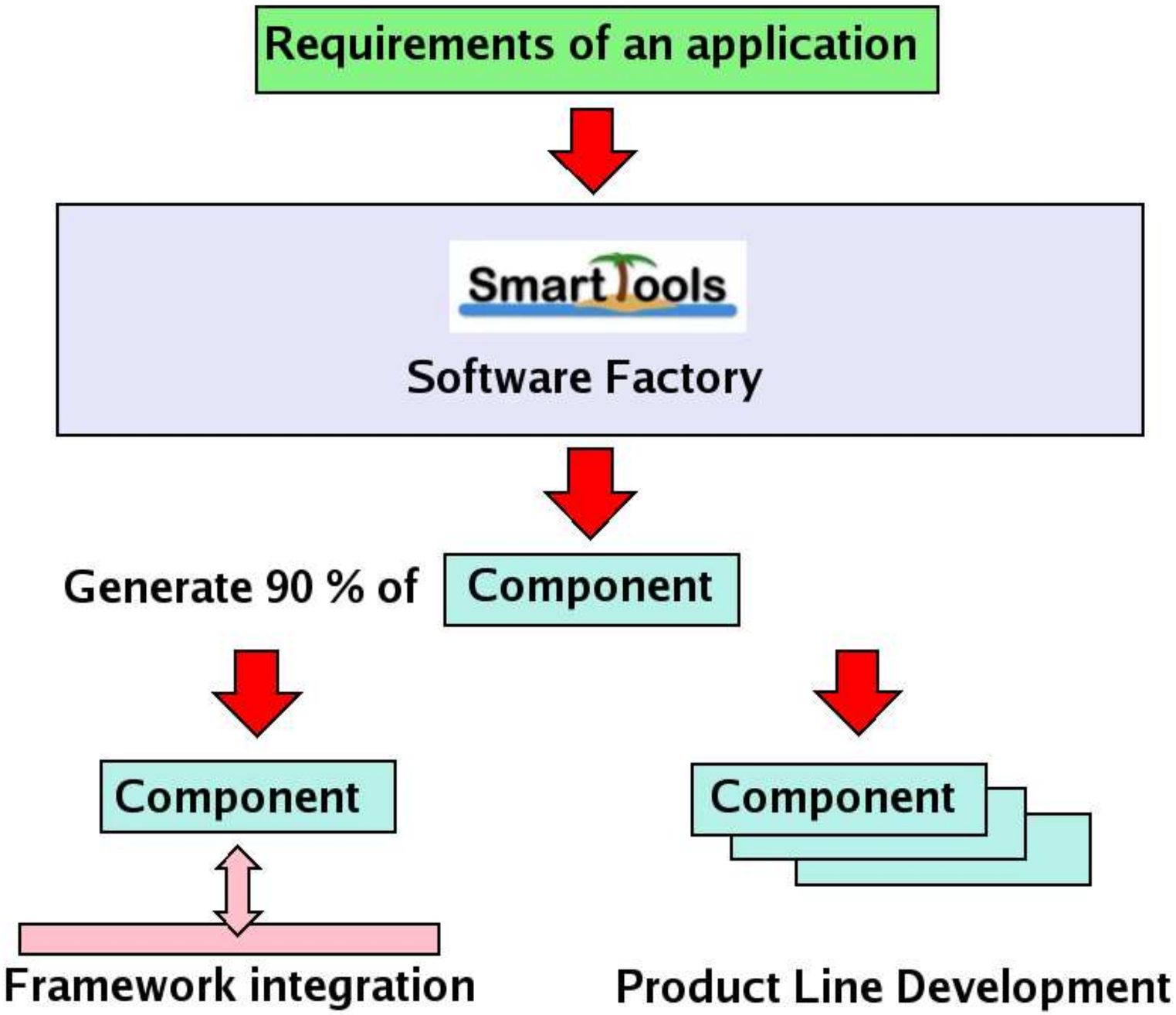
La programmation par modèles

par langages métiers (DSL)

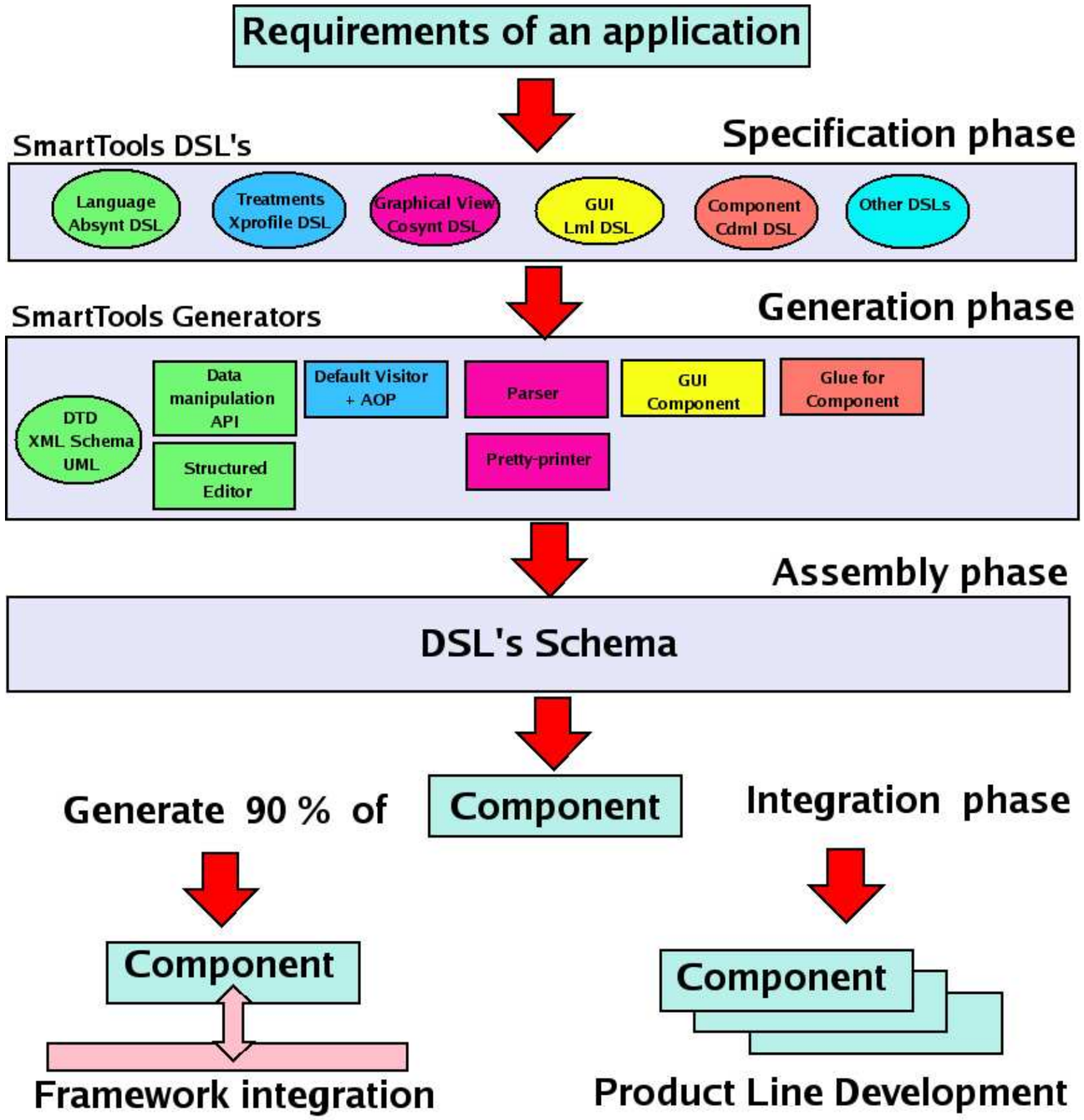
par transformations

Fertilisation croisée
entre divers
domaines de recherche

Aperçu de SmartTools



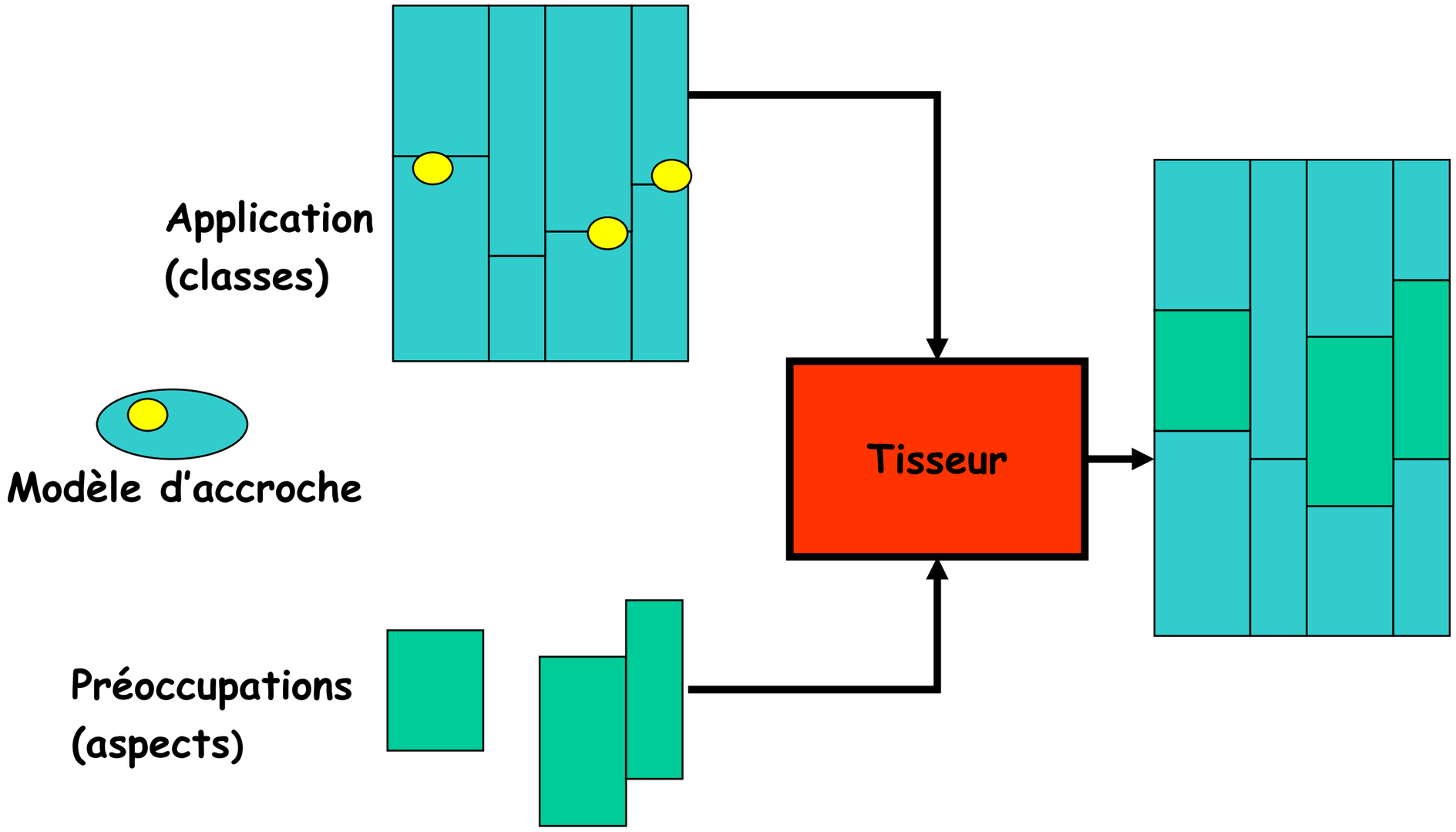
Aperçu de SmartTools



Programmation par aspects

Séparation des préoccupations

Modèle: Abstraction de l'application pour la préoccupation



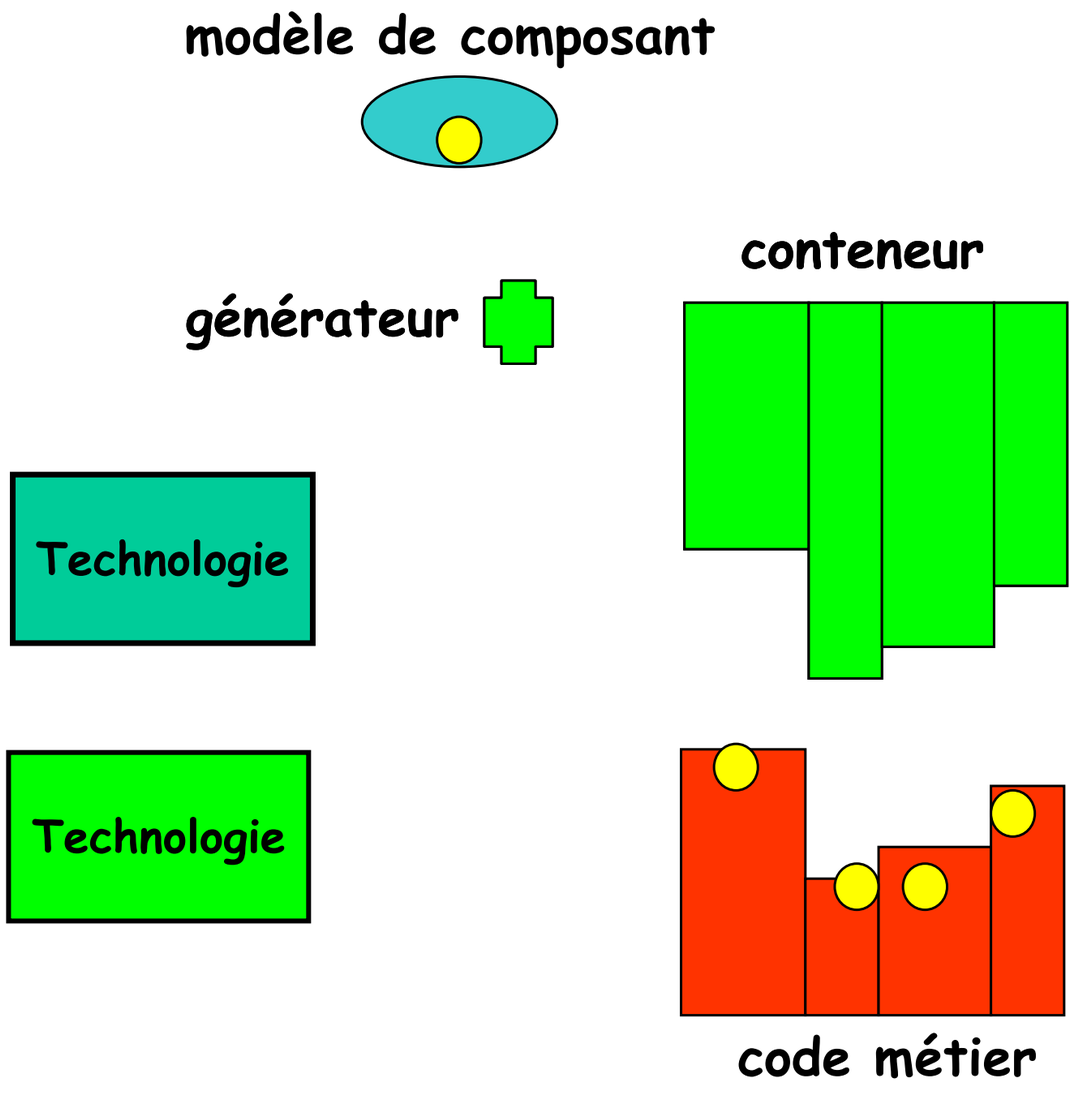
Programmation par Aspects

- Éclatement ou Morcellement du corps d'une méthode
- Nouvelles modularités orthogonales aux autres types de modularités
- Développement Incrémental
  - Assurer la correction ?
- Une réutilisation par transformation
  - Cette séparation permet une meilleure réutilisation à travers des transformations.

Programmation par composants

Séparation du code fonctionnel et du code non fonctionnel

Modèle : Abstraction du code métier pour le conteneur



Programmation par composants

- **Séparation** des préoccupations
  - L'appel d'une méthode transformée en quelque chose de beaucoup plus complexe

- **Abstraction** par rapport à un langage ou un environnement
  - Abstraction vis-à-vis d'une **API**

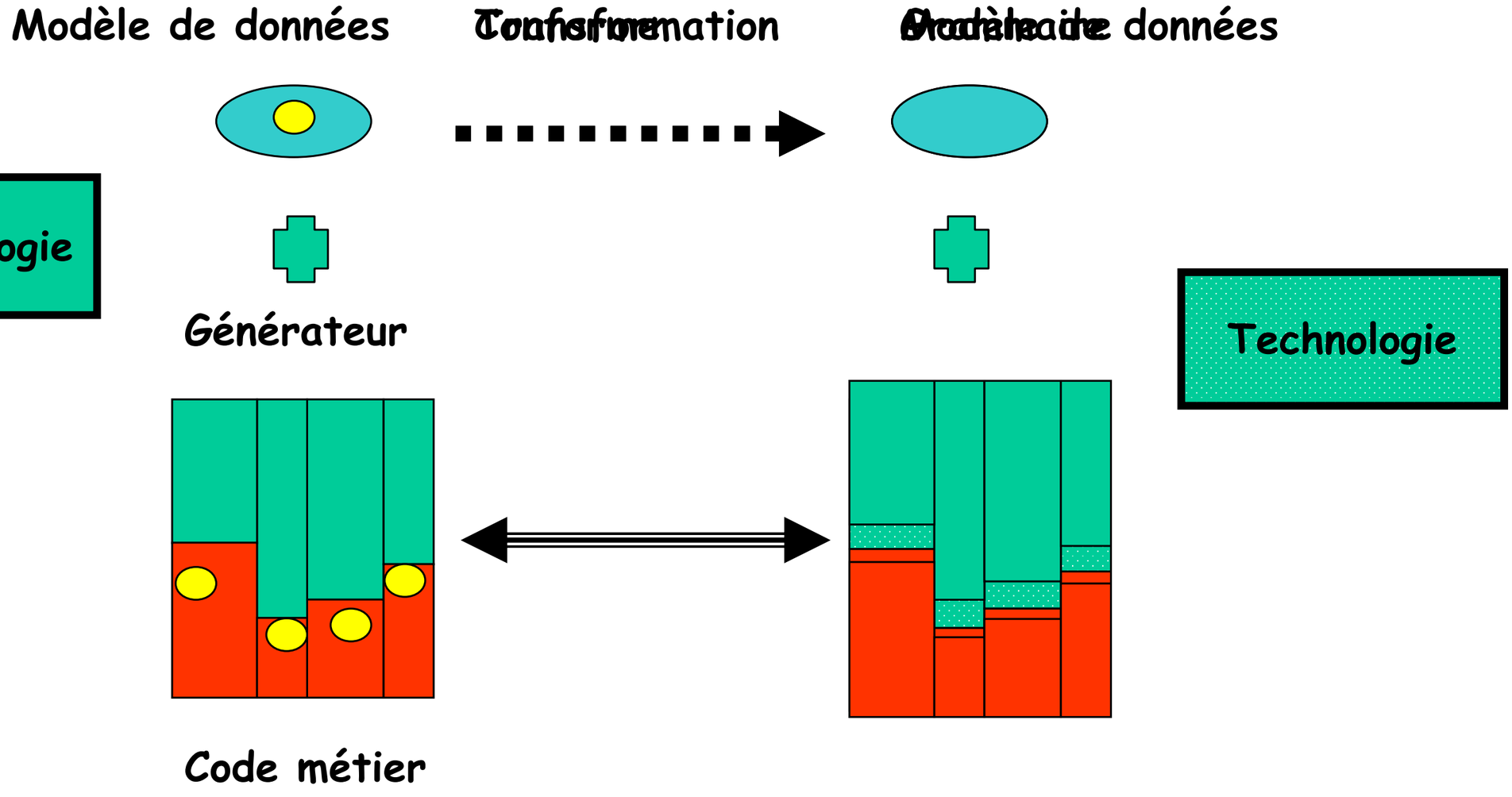
- Composant, entité logicielle beaucoup plus **autonome**
  - Architecture dynamique dirigée par les **services**

- Boite blanche, noire ou grise
  - **Exportation** d'un modèle de base.

Programmation par modèles, par DSLs

Indépendance du Modèle de données

Modèle : Abstraction du code métier pour les données



Programmation par modèles, par DSLs

- Définition (spécification) des structures de données
  - Divers implémentations et supports

- Programmation par modèles : par DSLs
  - Non limitée aux langages de programmation
  - Capture d'un savoir faire

- Spécification ou implémentation
- Données ou programmes (traitements)

- Transformation des traitements à travers la transformation du modèle

Continuité dans ma démarche: Résultats

- **Grammaire Attribuée: FNC-2**
  - Programmation par aspects
  - Programmation dirigée par les modèles = DSL
  - Programmation générative par excellence
- **Fabrique logicielle: SmartTools**
  - Approche beaucoup plus pragmatique
  - Programmation par DSL
  - Programmation générative
  - Programmation par aspects
  - Transformation de modèle (MDA)
  - Programmation par composants

Conclusion : développer Vite et Bien

- Vite
  - Patron de conception, DSL, programmation générative
  - Composant, Approche MDE, production en série

- Bien
  - Automatiser
  - Maintenance
  - Évolution
  - Ouverture
  - Plusieurs niveaux d'abstraction permettent une meilleure analyse des applications

- Cette approche va s'appliquer à beaucoup plus de domaines que l'on peut penser !

Au tout début d'un …

- Développement Incrémental (AOP).
- Développement par DSL.
- Développement par Transformation, surtout si on veut prendre en compte les traitements.
- Développement par composants adaptables et autonomes

- Fabrique Logicielle : Macro Programmation

- Les grands de l'informatique ont parfaitement identifié ces nouveaux besoins.

- Questions ?