A first experimentation on high-level tooling support upon Fractal

Frédéric Loiret\textsuperscript{1,2}, David Servat\textsuperscript{1}, Lionel Seinturier\textsuperscript{2}

\textsuperscript{1}CEA-LIST - L-LSP/Accord
\textsuperscript{2}INRIA Futurs - GOAL/Jacquard

5\textsuperscript{th} Fractal Workshop
PhD context

- **INRIA Jacquard Team**
  - Software components and AOP

- **CEA LLSP Team**
  - Model-based approaches (UML) to design real time and embedded systems

- **Component paradigm for RT**
  - Fractal component model extensions

- **Modelisation**
  - « High-level » tooling support

- **Execution**
  - Think implementation as a destination platform
First experiments

- **Meta-modelisation for Fractal architectural concepts**
  - Eclipse Modeling Framework overview
  - Fractal / Think ADL meta-model subset
  - Think V2 tools front end

- **Reverse engineering tool**
  - Source code analysis for Think components
  - Behavior extraction
Eclipse Modeling Framework overview

- A modeling and code generation framework for developing meta-model based applications and design tools

- A java implementation of the MOF (OMG) : EMF Ecore
  - class-relation constructs meta-model

- Code generation :
  - Java API (and classes) for model instance manipulations
  - Adapter classes (generated EMF model editor as Eclipse plugin)

- XML serialization for persistence support
  - (Model transformations)
Fractal / Think ADL meta-model (1)

- **EMF meta-model definition « from scratch »**
  - ADL and IDL Abstract Syntax Trees as MM
  - Hierarchy of interrelated concrete and abstract Ecore classes
  - Two types of associations:
    - containments
    - references

- **Packages organization**
Datatypes, Packages, IDL:

- **Datatypes**
  - `DataTypeDefinitionAbs`
  - `ComplexTypeAbs`
  - `PrimitiveDataTypeAbs`

- **Packages**
  - `PackageableElementAbs`

- **IDL**
  - `APILibrary`
  - `InterfaceAbs`

- **Operations**
  - `OperationSignature`

- **Parameter**
  - `Parameter`
  - `returnType`
  - `parameterType`

- **Interfaces**
  - `InterfaceSignature`
  - `RequiredInterface`
  - `ProvidedInterface`

- **Contingency**
  - `<contingency>`

- **Enums**
  - `ContingencyType`
Fractal / Think ADL meta-model (3)

- Simplified ADL meta-model:

  - **ComponentLibrary**
  - **PrimitiveComponent**
    - + isSingleton : boolean
    - + isTemplate : boolean
    - + lifeCycle : boolean
  - **CompositeComponent**
    - + controller : ControllerType
  - **ComponentAbs**
    - + compRef
    - + extends
  - **InterfaceAbs**
  - **Attribute**
  - **ProvidedInterface**
    - + serverItf
  - **RequiredInterface**
    - + bindAfterStart : boolean
    - + contingency : ContingencyType
  - **ExternalPropertyAbs**
    - + clientItf
    - + serverItf
  - **ProvidedDelegatedBinding**
    - + serverSubComp
  - **DelegatedBindingAbs**
    - + clientSubComp
  - **TypeDeclaration**
  - **InternalFeatureAbs**
    - + 1..*
  - **Eclipse Resource**
    - + skeleton source code
  - **SubComponent**
    - + clientSubComp
    - + serverSubComp
  - **BindingAbs**
    - + 1..*
  - **HorizontalBinding**

5th Fractal Workshop – 3/07/06, Nantes
Add element for a composite component
Think V2 tools front-end

- **Fractal/Think meta model**
- **Fractal/Think editor**
- **EMF model instances**
- **EMF code generation**
- **text2ecore**

**Eclipse plugin**

**IDL / ADL grammars**

**Think compilation chain**

**Kortex library**

**Projection / Synchronization**

**"Model world"**

**"Text world"**

**IDL / ADL AST's**

**Think V2 tools**

- **ADL parser**
- **IDL parser**

**Source files**

**ADL files**

**IDL files**
Discussions

- Homogeneous design environment for the user to specify Fractal applications
  - « Eclipse-based »

- Simpler for the user
  - Intuitive constructs using the editor
  - All meta-model concepts are « typed »
  - Semantical constraints based on meta-model structure
  - Resolv references during application specification

- Extensions
  - Graphical layer using GEF above EMF
  - Fractal EMF meta-model to UML2 profile
  - Expand the meta-model (≡ ADL extension)
    - Provide annotations mechanism for non-functional properties on model entities
    - Component behaviour
Toward a behaviour extraction and analysis

- **Inspired from SafeArchie [1]**
  - Behaviour contracts specified by I/O automata (messages traces exchanged by components with their environment).


- **Bottom-up approach**:
  - Extract behaviour from Think source code components
  - Integrate this behaviour to the meta-model
Reverse engineering tool chain

Think source code component

source code

AST

> - TranslationUnit
  · - NormalExternalDeclaration
  · - ADeclaration
    · - TypeSpecifierDeclarationSpecifier
      · - StructTypeSpecifier
        · - hellocalculus_lib_dispatchcompdata
          · - Declaration
            · - TypeSpecifierDeclarationSpecifier
              · - TypedefTypeSpecifier
                · - Calculus
              · - Declaration
                · - Pointer

Analysis

Think source code naming conventions

ADL definition

IDL definition

Analysis

I/O Synchronization

Meta-model integration

CompositeComponent

+ controller : ControllerType
+ isSingleton : boolean
+ isTemplate : boolean
+ lifeCycle : boolean

PrimitiveComponent

+ isSingleton : boolean
+ isTemplate : boolean
+ lifeCycle : boolean

BehaviourLTS

0..1

State

2..*

Transition

1..*

automata2ecore

Behaviour properties verification during user specification

5th Fractal Workshop – 3/07/06, Nantes