Auto-Fractal: (Ré-)assemblage dynamique et automatique d'applications à base de composants Fractal

G. Grondin\textsuperscript{1,2}, N. Bouraqadi\textsuperscript{1}, L. Vercouter\textsuperscript{2}

1: Dépt. IA - École des Mines de Douai
2: G2I/SMA - École des Mines de Saint-Étienne
Agenda

- Introduction
  - Motivating Example
  - Problem Statement

- Proposal
  - MaDcAr / Auto-Fractal
  - Example: Clock Application
    - Demo

- Conclusion
Introduction
Motivating Example

Loosing network connection

Performing a re-assembling
Problem Statement

- What is the problem?
  - The design of auto-adaptive Component-Based Applications (CBAs) is not easy (often ad hoc)
  - Adaptation = re-assembling of CBAs

- Issues related to the problem:
  - Automation of CBA’s adaptation
    - Minimizing impact on application’s administration
  - Dynamicity of CBA’s adaptation
    - Minimizing impact on application’s execution

- When is automatic and dynamic adaptation needed?
  - Contextual changes are unpredictable/frequent
  - Human interventions must be minimized
  ⇒ Ex: Ubiquitous Computing, Autonomic Computing, ...
Our Goal

Required properties

- **Abstraction**: independent of the used component model
- **Generality**: independent of the application domain
- **Openess**: customizability of the assembling process
- **Dynamicity**: re-assembling without stopping the whole application
- **Automation**: behaving on behalf of humans
- **Context-awareness**: behaving according to external events

**MaDcAr** : an abstract model for assembling engines

- Model for Automatic and Dynamic Component Assembly Reconfiguration
- [Grondin et al. CBSE 2006]
Auto-Fractal

An Assembling Engine for Fractal-Based Applications
Introduction to Auto-Fractal

- **Auto-Fractal** is the projection of MaDcAr abstract model on Fractal

- **Auto-Fractal** = **Assembling Engine** for Fractal:
  - **Inputs**: Components, Application’s Requirements
  - **Output**: a component assembly which meets these requirements
Assembling Engine

Context

Auto-Fractal

Assembling Engine

Constraint Solver

Application
(assembly)

Assembling policy

Components

Application
description
Application Description

Application description

Configuration

Role(min,max)

x-y-z

Role multiplicities

Required interface

Provided interface

Link: abstract
Description of a connection

Role: abstract
Description of a component

Initial values for the role’s attributes
Chat Client Application Description

Initial configuration

To use when losing network connection

To use when network connection is re-established while buffer is not empty
Re-Assembling Process

3 stages

- **Sense**
  - *When to re-assemble: Changes detection*

- **Plan**
  - *How to re-assemble: Decisions making*

- **Act**
  - *Perform the re-assembling accordingly*
Re-Assembling Process - Sense

- Context => set of sensors (software/hardware)
  - To automatically trigger (re-)assembling
    - Internal changes: components, configurations, …
    - External changes: location, network, neighborhood (soft/hard-ware), …
  - To collect data required when taking assembling decisions

⇒ Contextual situation = Set of sensor values within some range specified by the designer
Chat Client Application – Context

- **Triggering of assembling**
  - When the network connection status changes, or
  - When the buffer becomes empty

```
01 netWatch := NetworkWatcher new.
02 netWatch watchPeriod: 1000.
03 bufferWatch := BufferWatcher new.
04 bufferWatch watchPeriod: 500.
```

⇒ **Definition of two sensors**
  - netWatch (line 01)
  - bufferWatch (line 03)
Re-Assembling Process - Plan

- **Assembling policy** = a set of rules specifying the decisions to take for each contextual situation
  - **To choose a configuration**
    - **Identification of eligible configurations**
      - Contract testing for each pair \{role, component\}
      - Building of a « compatibility matrix »
    - **Configuration selection**
      - According to the « compatibility matrix » and the assembling policy
  - **To select components**
    - According to the chosen configuration and the assembling policy
  - **To schedule the re-assembling**
    - According to criteria such as assembling’s urgency, application’ services availability, minimization of the assembling’s cost, …
## Chat Client Application – Assembling Policy

### Configuration Selection

- Addition of several constraints on the new configuration (newConfig) according to the contextual situation

> Our constraint solver can infer an appropriate candidate for newConfig

```
05 (networkWatch isAvailable) ifFalse: [
06     constraintsSet add: [newConfig doesNotInclude: conferenceRole].
07     constraintsSet add: [newConfig includes: bufferRole].
08 ].
09 (networkWatch isAvailable) ifTrue: [
10     constraintsSet add: [newConfig includes: conferenceRole].
11     buffer := bufferRole getComponent.
12     (buffer isEmpty) ifFalse: [
13         constraintsSet add: [newConfig includes: bufferRole]].
14 ].
```

**Re-Assembling Process – Default Plan**

**Default configuration selection policy**
- If several configurations satisfy the assembling policy Then random selection of one of them
- If no configuration satisfies the assembling policy Then random selection of an eligible one

**Default component selection policy**
- After selecting a minimal number of components for each role, select the remaining components while roles are not maximally fulfilled

**Default application re-building policy**
- Minimize useless component replacements
- Minimize useless interfaces re-connections between components that are already in use
- Minimize the set of components that must be maintained in an idle state
- Minimize the delay of services unavailability for each component that must be maintained in an idle state
Re-Assembling Process – Act

Assembly (re-)building

Perform a sequence of re-assembling operations, according to decisions taken during the previous stage

- Disconnect two components
- Initialize a component
- Connect two components
- **Activate a component** (accept external requests)
- **Desactivate a component** (refuse external requests)
- **Buffer incoming requests** while a component is not available

Dynamicity
Demo Example

Clock application

(focusing on the assembling process)
Clock Application

User preferences have changed

Performing a re-assembling

#pm 11:59:23

Start Clock Stop Clock

Start Clock Stop Clock

#- 23:59:23
Clock Application – Application Description

Start Clock  Stop Clock

#- 23:59:23

Config1

Start Clock  Stop Clock

#pm 11:59:23

Config2
Clock Application – Application Description (1)

Config1

- H24(1,1) 0-23  increment
- Minutes(1,1) 0-59  increment
- Seconds(1,1) 0-59

Ticker(1,1)
Clock Application – Application Description (2)

Config2

Am/Pm(1,1)

H12(1,1)

Ticker(1,1)

Minutes(1,1)

Seconds(1,1)
**Clock Application Description**

The user prefers the 24H Time System.

The user prefers the Am/Pm Time System.
Clock Application – Context

- Triggering of assembling
  - When the user manually modifies its preferences

01 preferenceWatch := PreferenceWatcher new.

⇒ Definition of one sensor
  - preferenceWatch (line 01)
Clock Application – Assembling Policy

Configuration Selection

- Addition of several constraints on the new configuration (newConfig) according to the contextual situation

⇒ Our constraint solver can infer an appropriate candidate for newConfig

```plaintext
02 (preferenceWatch isSetTo24HoursTimeSystem)
03 ifTrue: [
04    constraintSet add:
05        (newConfig doesNotIncludes: roleAmPm)]
06 ifFalse: [
07    constraintSet add:
08        (newConfig includes: roleAmPm)].
```
DEMO
Conclusion
Advantages of Auto-Fractal

- **Specific and Configurable assembling policies**
  - Allows global control over the assembling process

- **Total automation of adaptations**
  - Allows total control over adaptation’s triggering, decision and realization steps

- **Separated descriptions of the application’s functionalities and the application’s adaptations**
  - Eases understanding, reuse and evolution of applications

- **Low coupling between configurations and components**
  - Allows configurations’ reuse

- **Support of unpredicted dynamic adaptations**
  - Allows dealing with context-sensitive applications (Ubiquitous Computing, Autonomic Computing, …)
Ongoing work

- **First implementation of Auto-Fractal for FracTalk**
  - FracTalk = our implementation of the Fractal component model (in Smalltalk)

- **Support for dynamic assembling**
  - State transfer when replacing a component
  - Components inter-dependencies

- **Support for automatic detection of contextual changes**
MaDcAr : model for component assembling
- Abstraction
- Dynamicity
- Automation

Auto-Fractal : MaDcAr for Fractal components

http://csl.ensm-douai.fr/grondin

This work is supported by the CPER TAC of the region Nord-Pas de Calais and the european fund FEDER
State of Art - Related Work
Related Work (1) - ADL

- C2 [Medvidovic96]
  - Description
    - ADL allowing dynamic modification of an architecture
    - Adaptative policies based on connectors
  - Limitations
    - Rules are locally defined in each connectors
      - Inconsistencies are detected a posteriori
    - Adaptation must be triggered manually
    - Not possible to specify when or under what condition configuration are to be carried out

⇒ A high degree of control is needed to automated re-assembling
**Related Work (2) - CBSE**

- SAFRAN [Dav05]
  - **Description**
    - Creation of dynamically reconfigurable component
    - Adaptative policies based on Event-Condition-Action rules (ECA)
  - **Limitations**
    - Rules are locally defined in each component
      - Inconsistencies are detected a posteriori
      - Unpredicted cost of the adaptations
    - Adaptation concern strongly coupled with the architecture
      - Need to introduce an artificial composite to share a policy between components
    - Building an application from scratch is not addressed

⇒ High level concepts are needed to be independent of a particular component model
Related Work (3) - Other

- **ADL**
  - Darwin [Imperial College London]
    - (+) dynamic instanciation rules for architecture creation
    - (-) re-assembling not addressed
  - Rapide [Carnegie Mellon University]
    - (+) declaration of interconnection rules
    - (-) used only for architecture verification (not architecture assembling) like Wright

- **CBSE**
  - SOFA [Charles University]
    - (+/-) Dynamic –but partial– adaptation of hierarchical component
  - CASA [Zurich University]
    - (+) Automatic and dynamic application adaptation
    - (-) Based on a specific component model and implementation