Automated and Unanticipated Flexible Component Substitution

Nicolas Desnos[1] Marianne Huchard[2]
Christelle Urtado[1] Sylvain Vauttier[1] Guy Tremblay[3]

[1]LGI2P - Ecole des Mines d'Alès
Nîmes - France
nicolas.desnos@ema.fr, christelle.urtado@ema.fr, sylvain.vauttier@ema.fr

[2]LIRMM – CNRS
Montpellier - France
huchard@lirmm.fr

[3]Département informatique, UQAM, Montréal, QC, Canada
tremblay.guy@uqam.ca

Outline

Context and Definition

Context

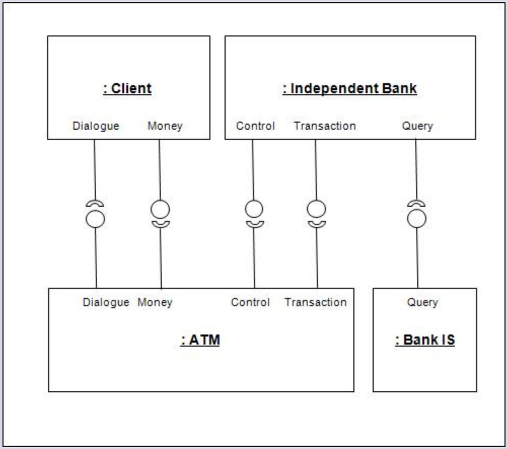
- Dynamic evolution of software systems
- Component based development

Key concepts

- Component
    - Black box view = Provisions and requirements
    - White box view = Implementation
- Architecture
    - Component classes, connections
    - Functional and non-functional requirements
- Component assembly
    - Architecture instantiation
- Validity = Correctness (syntactic/semantic) and completeness (connections satisfying requirements)

Desnos, Huchard, Urtado, Vauttier, Tremblay

Example of component assembly



Desnos, Huchard, Urtado, Vauttier, Tremblay

Context and Motivation

Why replace components ?

- Obsolescence, failure, unavailability, etc.

In which environments ?

- Distributed and ubiquitous computing
- Mobile computing

Issue

- How can a component be replaced in a safe and unanticipated way ?
  - Safe. . . with respect to functional requirements
    $\Rightarrow$ Functional objectives must still be reached

Existing Solutions

Context

- Input = A valid component assembly that conforms to its architecture [Wright]
- Problem = How to replace a component in safe way ?

Typical solution

- Component to component substitution
- The new component
  - can provide more services
  - can require less services

Drawback

- What should be done when no single component is available to replace the target component ?

Flexible Component Substitution Using an Automatic Building Process

Goal

- Replace a target component

Two possible cases

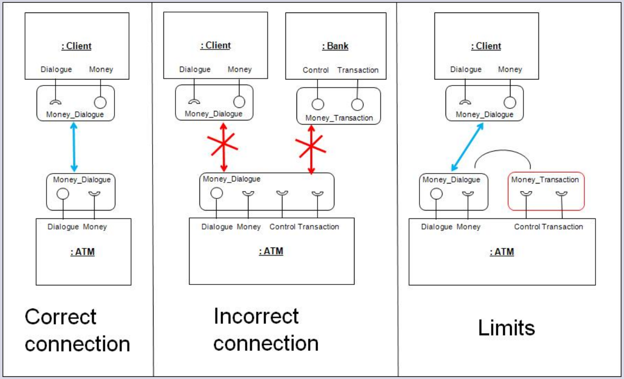
- There is a unique substitutable component $\Rightarrow$ Substitute component by the candidate component
- There are no candidates $\Rightarrow$ Substitute component by an assembly while preserving validity

Key elements of our approach

- Primitive and composite port model
- Rebuilding process

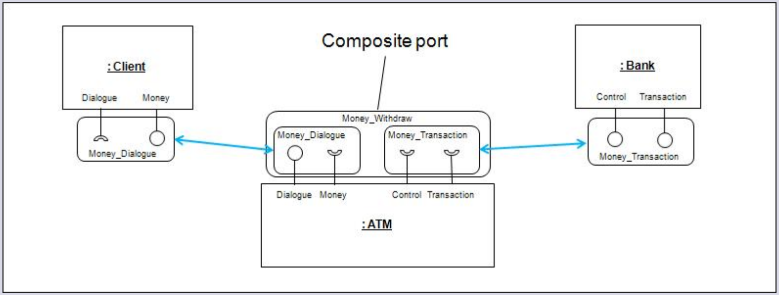
Primitive Port

- Combines a set of interfaces [*à la* UML 2.0]
- Peer-to-peer and atomic connection
- Limit : Cannot express multi-peer connection



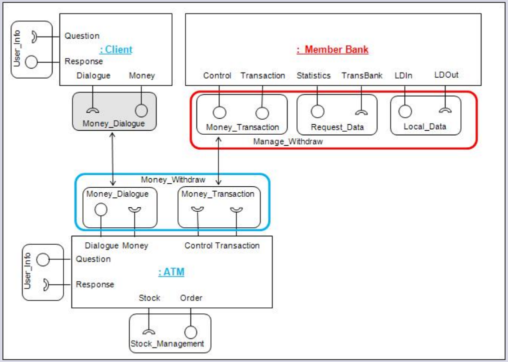
Composite Port

- Combines a set of ports—primitive or composite
- Multi-peer non-atomic connection



Completeness (1/2)

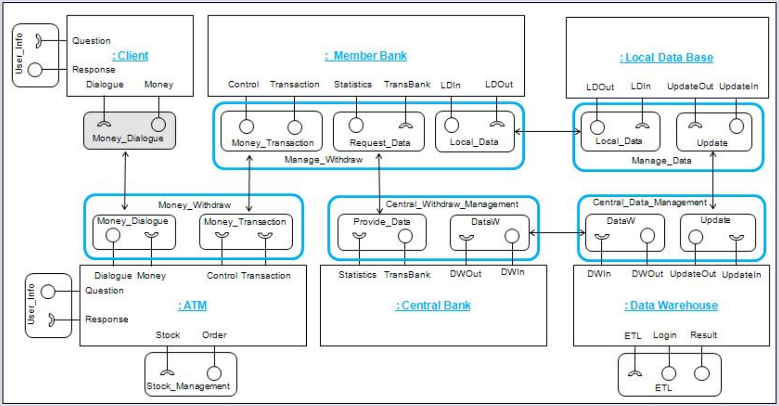
- A **component** is **coherent** if all its composite ports are coherent
- A **composite port** is **coherent** if all its primitive ports are either connected or disconnected



Desnos, Huchard, Urtado, Vauttier, Tremblay

Completeness (2/2)

- An **architecture** is **complete** if all its components are coherent and if each primitive port which represents a functional objective is connected

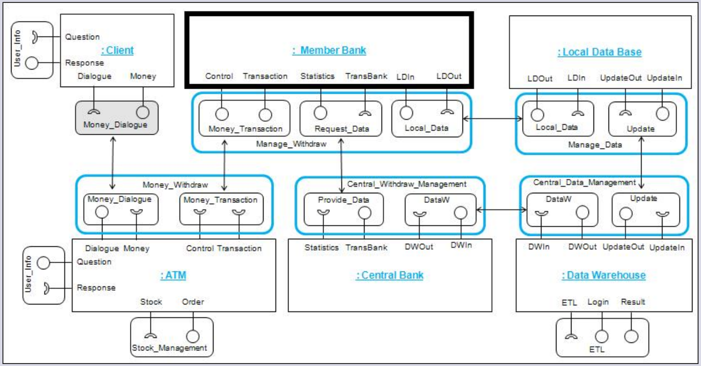


Desnos, Huchard, Urtado, Vauttier, Tremblay

Component Replacement process

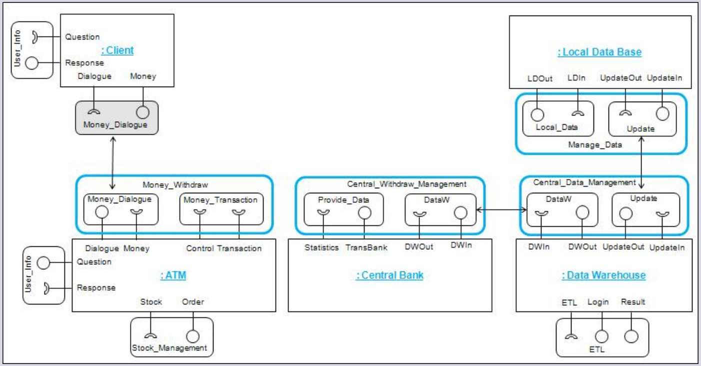
1. Remove the target component
2. Remove the resulting dead components
3. Re-build a complete assembly using bottom-up process (previous work)
4. Check the correctness of suggested assembly

Step 1 : Remove the target component



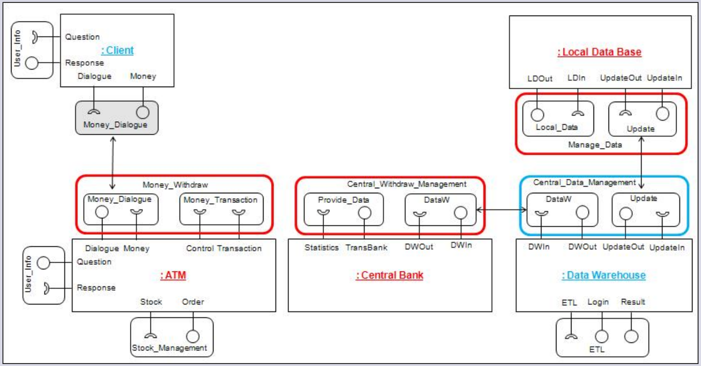
- Select the target component ⇒ `Member Bank` component is selected

Step 1 : Remove the target component



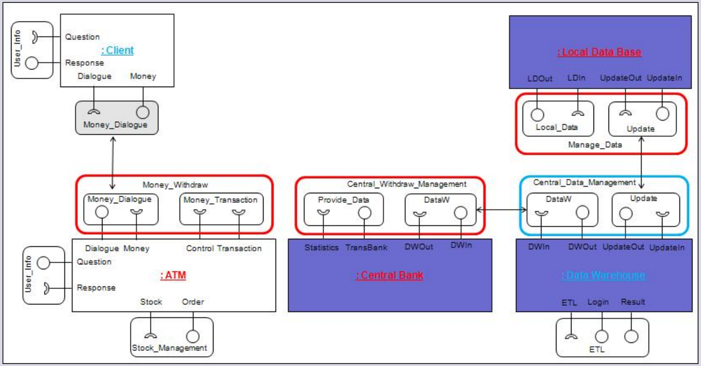
- Remove target component ⇒ `Member Bank` component is removed

Step 1 : Remove the target component



- Completeness is lost ⇒
  Some composite ports become non-coherent

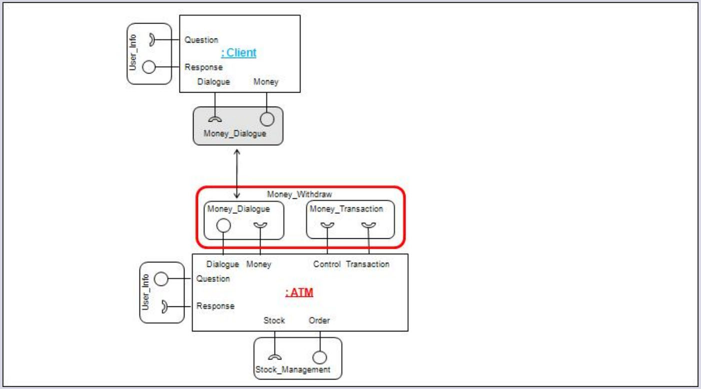
Step 2 : Remove the dead components



- Completeness is lost
  Some component are dead

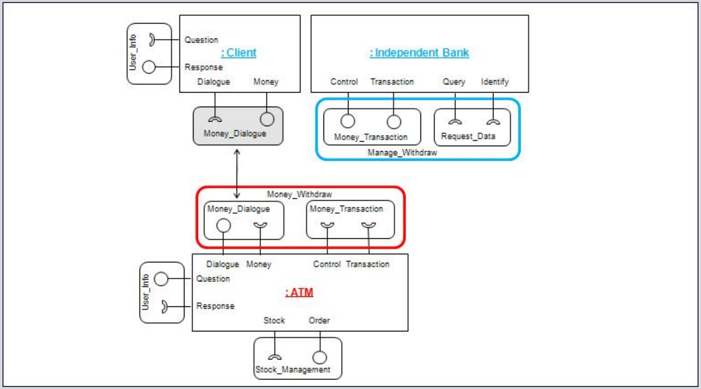
Desnos, Huchard, Urtado, Vauttier, Tremblay

Step 3 : Rebuild a complete assembly



- Rebuild the assembly starting from the **live components**

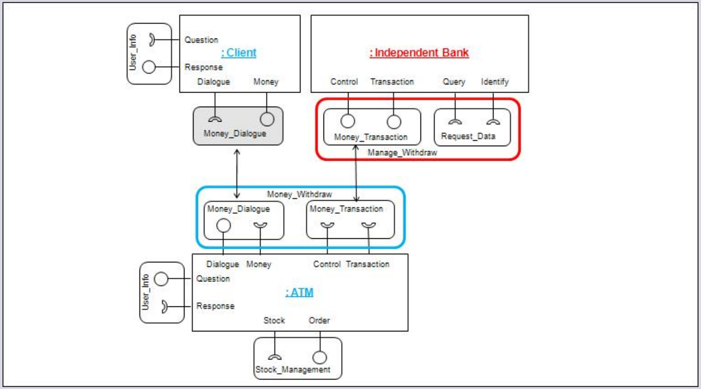
Step 3 : Rebuild a complete assembly



- Run our incremental algorithm [EWSA-06]

Desnos, Huchard, Urtado, Vauttier, Tremblay

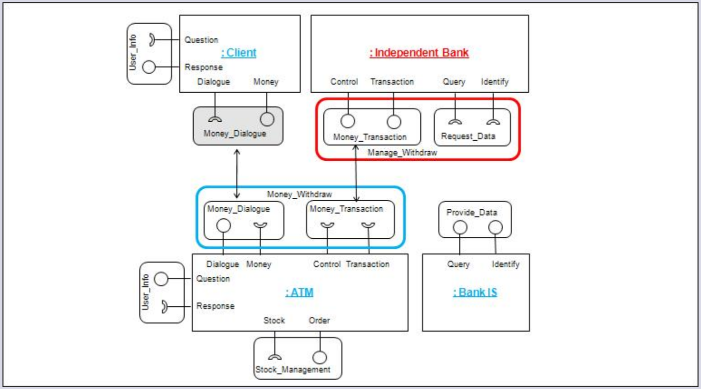
Step 3 : Rebuild a complete assembly



- Run our incremental algorithm [EWSA-06]

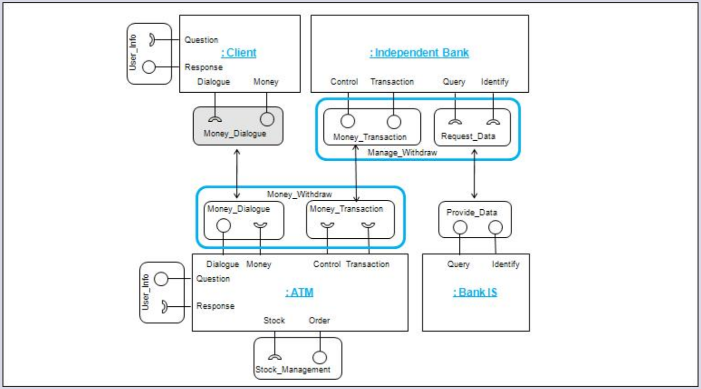
Desnos, Huchard, Urtado, Vauttier, Tremblay

Step 3 : Rebuild a complete assembly



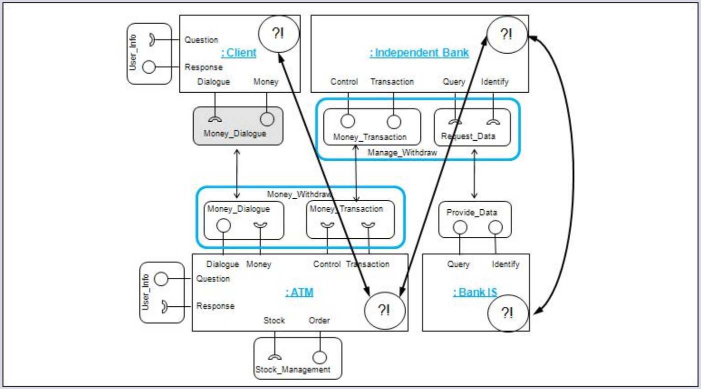
- Run our incremental algorithm [EWSA-06]

Step 3 : Rebuild a complete assembly



- The resulting component assembly is complete

Step 4 : Check correctness



- Correctness is checked
- Done using existing work [SOFA project]

Implementation and experiments

Implementation

- Based on Julia (Fractal component model)
  - `http ://fractal.objectweb.org`
- Parts already implemented
  - Port meta-model
  - Building and re-building algorithms

Experiments

- Component assembly can be rebuilt in :
  - 80% of the cases with our approach
  - 19% of the cases with only component-to-component substitution
- Dead components are reused in only 20% of the cases

Conclusion and Future Work

Contribution

- Innovative solution to dynamic component substitution when there are no candidate for component-to-component substitution
  - Replace a component by a component assembly

    . . . in an automatic way
  - Remove dead components
  - Guarantee validity
- Increase the probability to find a valid architecture
- Implementation exists in Fractal component model

Future Work

- Integrate this tool in a more general framework
- Use this work in Mobile & Ubiquitous computing
- Generate ports from protocols

Desnos, Huchard, Urtado, Vauttier, Tremblay