

A Reverse Engineering Tool for Precise Class Diagrams

Yann-Gaël Guéhéneuc
guehene@iro.umontreal.ca





Statements

- Maintenance amounts—at least—for 50% of the total cost of a program
- Maintainers spend—at least—50% of their time understanding source code
- ❖ Any help in understanding a program would reduce the cost



Context

- UML is a *de facto* standard
- Developers use UML-like class diagrams intensively
- Class diagrams
 - Classes, interfaces
 - specialisation, instantiation, use, association, aggregation, composition



Problem

- Code is the only source of information
- Maintainers would benefit from class diagrams to understand programs
- But, class diagrams
 - *Often* are obsolete and imprecise
 - Do not reflect the *real* implementation and behaviour



Problem (cont'd)

- Maintainers have tools, but these tools recover only the simplest constituents of UML-like class diagrams
- Recovered class diagrams
 - Not precise
 - Because of the lack of definitions at design- and implementation-level



Solution

■ Study

- Definitions of UML constituents
- Recovery of UML constituents in object-oriented source code

❖ A reverse engineering tool for precise class diagrams



Definitions

- **Classes (class-based languages)**
 - “Matrices” for instances
 - Inner states and public services
- **Interfaces**
 - Types
 - Services that instances must provide



Definitions

■ Specialisation

- Relationship between a set of similar entities and another entity* that contains the common aspects of those entities

Definitions

- Use, association, aggregation, composition
 - Up to now, no consensual definitions
 - Link (message send), relationship

Link		
<i>Origin</i>	<i>Means</i>	<i>Target</i>
Entity	Direct / Field	Entity
Instance	Direct	Instance
Instance	Field	Instance
Instance	Field + Lifetime property	Instance

Is described by

Relationship		
<i>Origin</i>	<i>Name</i>	<i>Target</i>
Entity	Use	Entity
Entity	Association	Entity
Class	Aggregation	Entity
Class	Composition	Entity

Recovery

- Classes
- Interfaces
- Specialisation
 - Exist syntactically in the source code

```
public interface I {  
    ...  
}  
  
public class C implements I {  
    ...  
}
```



Recovery

- Use, association, aggregation, composition
 - Do not exist syntactically in the source code
 - Expressed with four minimal properties of the relationships (cf. p. 8)

Recovery

- Use, association, aggregation, composition
 - Exclusivity, $EX(A, B) \in \{\text{true}, \text{false}\}$
 - Invocation site, $IS(A, B) \subseteq \{\text{field}, \text{parameter}, \text{local variable}, \dots\}$
 - Lifetime, $LT(A, B) \in \{+, -\}$
 - Multiplicity, $MU(A, B) \subset \mathbb{N} \cup \{+\infty\}$



Recovery

- Use, association, aggregation, composition
 - Rewriting relationships as conjunctions of the four properties (cf. p. 8-9)
 - $US(A, B)$
 - $AS(A, B)$
 - $AG(A, B)$
 - $CO(A, B)$

Recovery

- Use, association, aggregation, composition

- Example

- $AS(A, B) =$

$$\begin{aligned} & (IS(A, B) \subseteq \{\text{field, parameter, ...}\}) \wedge (IS(B, A) = \emptyset) \wedge \\ & (EX(A, B) \in \{\text{true, false}\}) \wedge (EX(B, A) \in \{\text{true, false}\}) \wedge \\ & (LT(A, B) \in \{+, -\}) \wedge (LT(B, A) \in \{+, -\}) \wedge \\ & (MU(A, B) = [0, +\infty]) \wedge (MU(B, A) = [0, +\infty]) \end{aligned}$$

- Order among the relationships



Recovery

■ Algorithms

- Class, interface, specialisation
 - Syntactic analyses

- Use, association, aggregation, composition
 - Computation for each class of the properties with respect to other classes
 - Order among the relationships
 - Dynamic analyses



Conclusion

■ Before

- Maintainers have tools, but these tools recover only the simplest constituents of UML-like class diagrams

■ Now

- Definitions of UML constituents
- Recovery of UML constituents
- A tool for precise class diagrams



Future work

- Use of static analyses instead of dynamic analyses
- Recovery of other UML constituents
 - Data types, implementation classes...
- Development of layout algorithms for UML-like class diagrams
- Experimental validation of the recovered UML-like class diagrams



Thanks

Questions?

Comments?