Reverse Engineering Method Stereotypes

Software Development Laboratory
<SDML>

Natalia Dragan
Michael L. Collard
Jonathan I. Maletic

KENT STATE UNIVERSITY

<SDML>

Background

- Stereotypes are a mechanism (in UML) to extend and define semantics
- Simple documentation technique for methods
  - Role and semantics of the method
  - Common vocabulary (well known terms)
  - Predicate, mutator, accessor
- However, method stereotypes are rarely explicitly documented by developers

KENT STATE UNIVERSITY

<SDML>

Motivation

- Support comprehension of method/class
- Changes to a method that change its stereotype is an indication of design change
- LCOM metrics are biased by certain types of methods (accessor & constructors)
- Good object abstraction requires good method abstraction
- Class stereotypes (e.g., boundary, control, entity) are based on the methods

KENT STATE UNIVERSITY

<SDML>

Design Recovery

- Reverse engineering method stereotypes supports higher-level design recovery of object-oriented systems
- Little research has been conducted on method stereotypes in the context of reverse engineering or automatic identification

KENT STATE UNIVERSITY

<SDML>

Objective

- Define a taxonomy of method stereotypes
- Automatically identify a method's stereotype
- Re-document the method (annotate with a comment in the source)

- Scalable to large systems

KENT STATE UNIVERSITY

<SDML>

Developing a Taxonomy

- Examined previous classifications
- Little work on the topic
- No formal in-depth studies
- Forward engineering view point

KENT STATE UNIVERSITY

<SDML>

Research on Stereotypes

- Method classifications with respect to the internal state of objects at the design level

  [Fowler00], [Arevalo03]

  - Focused on how a method accesses its data members rather than a view on the primary purpose of the method
- Method categorization with respect to collaborations between methods [Lanza01], [Clarke03]
  - Limited to generalization relationships
- Method stereotypes for assisting in program development of C++ and Java [Workman02], [Riehle01]
  - No means for detection are given

KENT STATE UNIVERSITY

<SDML>

Method Stereotypes

- Behavioral characteristic
  - Structural
    - Provide/support the structure of the class
  - Collaborational
    - Communication between different objects
  - Creational
    - Create/destroy objects
- Type of data access to data members
  - Read (accessors)
  - Write (mutators)

KENT STATE UNIVERSITY

<SDML>

Stereotype Taxonomy

Structural		Collaborational	Creational
Accessor	**Mutator**		
Get	Set	Collaborator-Accessor	Constructor
Predicate	Command	Collaborator-Mutator	Copy Constructor
Property			Destructor
			Factory

KENT STATE UNIVERSITY

<SDML>

Structural Methods: Accessors

- A ***get*** method is an accessor that returns the value of a data member

```
const string& getName() const;
```

- A ***predicate*** method is an accessor that returns a Boolean result

```
bool isEmpty() const;
```

- A ***property*** method is an accessor that returns information about an object based on data member values

```
int indexOfMinElement (int index) const;
```

KENT STATE UNIVERSITY

<SDML>

Structural Methods: Mutators

- A ***set*** method is a mutator that changes the value of a data member

```
void setName(const string& name);
```

- A ***command*** is a mutator that executes a complex change of the object's state
  - The change may involve several data members
  - May change the data members either directly or indirectly using another mutator

```
void draw(int x, int y);
```

KENT STATE UNIVERSITY

<SDML>

Collaborational Methods

- A ***collaborator*** is a method that works other objects (different from its own class)
  - Parameters that are objects
  - Local objects
  - Objects accessed indirectly through a data member (e.g., a vector of object pointers)

```
bool hasControlPoints
(const PlotterBase*plotter) const;
```

KENT STATE UNIVERSITY

<SDML>

Creational Methods: Factory

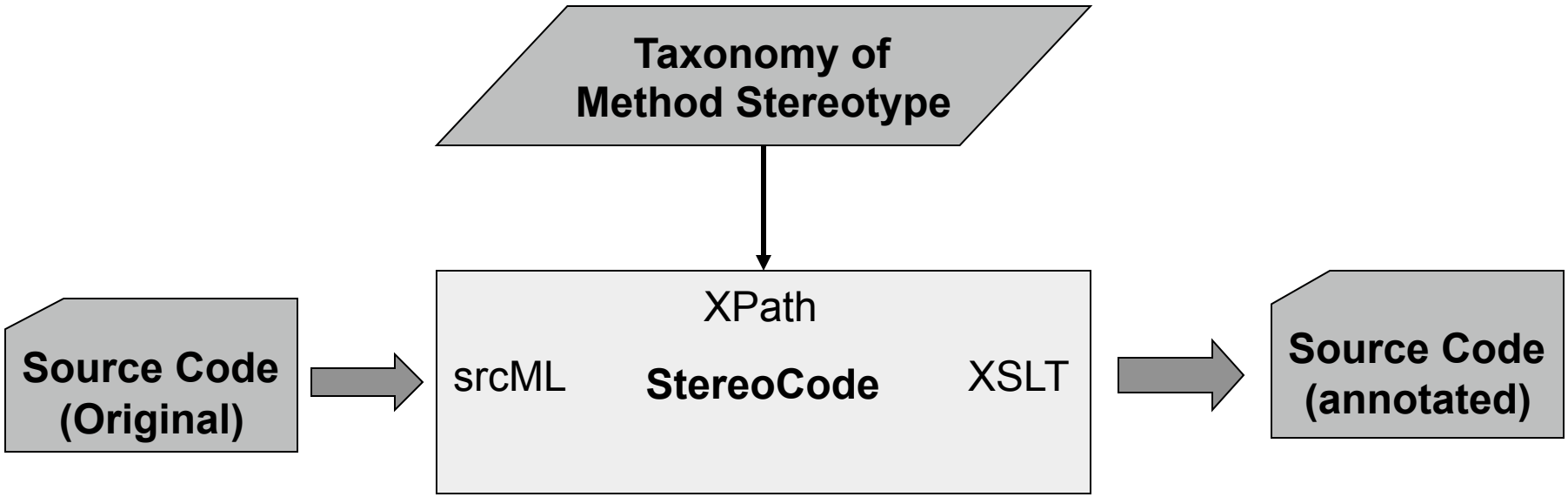
- A ***factory*** method is one that creates an object and returns it to the client (object creation method)
  - Factory methods work outside of the class and change the state of the external objects with which they have relations

```
PlotterBase* createDisplay
                    (const string& name);
```

KENT STATE UNIVERSITY

<SDML>

Approach



KENT STATE UNIVERSITY

<SDML>

Implementation for C++

- Need to analyze:
  - Access to data members
  - Return type
  - Parameter type
  - Const-ness
  - Type of local variables

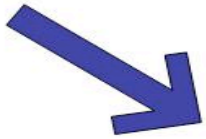
KENT STATE UNIVERSITY

<SDML>

Example

STARTING POINT

```
class  DataSource
{
private:
  string m_ds_name;
  vector<string> m_labels;
public:
  const string& getName() const;
  bool isValidLabel(const string& label) const;
  void setLabels(const vector<string>& v);
};
```



RESULT

```
class  DataSource :public Observable
{
private:
  string m_ds_name;
  vector<string> m_labels;
public:
  /** @stereotype get */
  const string& getName() const;

  /** @stereotype predicate */
  bool isValidLabel(const string& label) const;

  /** @stereotype set */
  void setLabels(const vector<string>& v);
};
```

KENT STATE UNIVERSITY

<SDML>

Evaluation

- Tools for stereotypes detection were applied to *HippoDraw* and *Qt*
- Each method was automatically labeled with a stereotype and the original source code was re-documented by our tools
- To assess the approach and the tools an experienced developer rated the automatically identified annotations in *HippoDraw*

KENT STATE UNIVERSITY

<SDML>

HippoDraw and *Qt*

- *HippoDraw* - open-source application providing a data-analysis environment
  - 60 KLOC of source code in over 400 C++ files (200 classes with 2900 methods & free functions)
- *Qt* - cross-platform C++ GUI framework
  - 4.1.2. version - about 1000 KLOC (over 1000 classes with 20900 methods)
- The source for both is well written and follows a consistent object oriented style

KENT STATE UNIVERSITY

<SDML>

Method Classifications

Stereotype(s)	Occurrences		%	
	Hippo	Qt	Hippo	Qt
Number of methods labeled with only one stereotype	1357	6410	50.1	30.7
Number of methods labeled with two stereotypes	1099	14067	40.6	67.4
unclassified	220	386	8.1	1.8
empty_method	30	8	1.1	0.04
Overall Total	2706	20869	100	100

KENT STATE UNIVERSITY

<SDML>

Single Stereotypes

Stereotype(s)	Occurrences		%	
	Hippo	Qt	Hippo	Qt
command	439	1281	16.2	6.1
property	361	1098	13.3	5.3
collaborator	239	3707	8.8	17.7
get	133	109	4.9	0.5
predicate	99	54	3.7	0.3
set	84	161	3.1	0.8
factory	2		0.1	
Number of methods labeled with only one stereotype	1357	6410	50.1	30.7

KENT STATE UNIVERSITY

<SDML>

Multiple Stereotypes

Stereotype(s)	Occurrences		%	
	Hippo	Qt	Hippo	Qt
collaborator,command	623	8546	23.0	40.9
collaborator,factory	296	889	10.9	4.3
collaborator,property	90	2806	3.3	13.4
collaborator,set	30	819	1.1	3.9
collaborator,predicate	23	471	0.8	2.3
collaborator,get	22	378	0.8	1.8
collaborator,empty_method	14	156	0.5	0.8
property,empty_method	1		0.0	
Number of methods labeled with two stereotypes	1099	14067	40.6	67.4

KENT STATE UNIVERSITY

<SDML>

Assessment Methodology

- A subset of 19 classes and 365 methods (~14% of the system) with a wide diversity of stereotypes was selected from *Hippodraw*
- A subject (an experienced developer) was given the taxonomy with the definitions of each stereotype
- Rating of a method's stereotype on a Likert scale: *Very Good*, *Good*, *Fair*, and *Poor*

KENT STATE UNIVERSITY

<SDML>

Subject Assessment

Hippodraw	Very good	Good	Fair	Poor	Total
Subject's Assessment	289 (79%)	40 (11%)	15 (4%)	21 (6%)	365 (100%)
Errors due to poor design of methods		1		5	6 (1.6%)
Errors due to incomplete analysis		19		5	24 (6.6%)
Errors due to differences in interpretation		7	5	11	23 (6.3%)

90% **10%**

KENT STATE UNIVERSITY

<SDML>

Scalability

- Translating to srcML
  - ~15 seconds for *HippoDraw*
  - ~3 minutes for *Qt*
- The identification and annotation
  - ~30 seconds for *HippoDraw*
  - ~6 minutes for *Qt*
- Converting back to raw C++
  - ~2 seconds for *HippoDraw*
  - ~30 seconds for *Qt*

KENT STATE UNIVERSITY

<SDML>

Contributions

- Defined a taxonomy of method stereotypes and rules for the automatic identification in C++
- Developed a tool, *StereoCode*, to automatically identify the stereotypes and re-document source code
- Evaluation servers as a benchmark for further studies and investigations

KENT STATE UNIVERSITY

<SDML>

Conclusions

- Our stereotype classification along with the tool *StereoCode* for automatically identifying and re-documenting method stereotypes is sound and efficient
- Our results were very good as an experienced developer agreed 90% of the time with our classification
- *StereoCode* tool, based on a lightweight static program analysis approach, is efficient and usable

KENT STATE UNIVERSITY

<SDML>

Future work

- Construct design-quality metrics based on stereotype classification
- Extend this approach to automatically reverse engineer class stereotypes