ESUG 2003

Induced Intentional Software Views

Tom Tourwé  Johan Brichau
Andy Kellens and Kris Gybels

Programming Technology Lab
Department of Computer Science
Vrije Universiteit Brussel
Problems with Software Documentation

- Software becomes very large, more complex and constantly evolves
- Software documentation is extremely important to cope with these issues
  - avoid design degradation
  - understand inner workings
  - implement correct behaviour
- Documentation is often non existent or outdated
  - not active part of development process
  - documentation and implementation are separated
  - not robust w.r.t. evolution
Software Views

- Documentation technique used to highlight important design structures
  - design patterns, framework hotspots, collaborations, ...
- Collection of source code artifacts
  - classes, methods, variables, ...
- Two different kinds of software views
  - Extensional views
  - Intentional views
Extensional Views

- **Manual enumeration** of source code artifacts

**Advantages**
- easy to define (drag & drop)

**Disadvantages**
- not robust w.r.t. evolution
- not scalable
- not intention revealing

Reduces interest of using software views
Intentional Views

Star Browser on: printOn: methods

Name: printOn: methods

Description:

[Soul allClasses select: [:t1 | t1 selectors includes: #printOn:]]
Intentional Views

- Defined by means of an intentional description
  - executable expression in a programming language
  - view’s content is computed from the source code

Advantages
- robust w.r.t. evolution
- scalable
- intention revealing

Complicates use of Intentional Views

Disadvantages
- hard to define (requires meta-programming skills)
- risk to be overly general
- requires detailed knowledge of the application’s internal structure
Induced Intentional Views

- Combines advantages of extensional and intentional views
  - ease of use of extensional views
  - robustness and scalability of intentional views

- Inducing views
  - manually classify source code artifacts
  - automatically derive intention behind it

- Techniques
  - Logic Metaprogramming
    - to connect views to implementation
  - Inductive Logic Programming (Machine Learning)
    - to derive intention automatically
Logic Meta Programming (LMP)

- Using a logic programming language (Prolog) at the meta level to reason about and manipulate programs at the base level (in Smalltalk)
- Allows to define intentional views in a concise and declarative manner
- **SOUL**
  - Interpreter integrated in VW
  - Contains extensive library of logic predicates that consult source code
## Inductive Logic Programming

- **Machine learning technique**
  - Discovers a general pattern underlying a number of examples
  - Requires a set of examples and a background theory

<table>
<thead>
<tr>
<th>Examples</th>
<th>Background Theory</th>
<th>Induced Logic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>grandFather(tom,bob).</td>
<td>father(tom,peter).</td>
<td>grandFather(?grandfather,?person) if father(?grandfather,?father),</td>
</tr>
<tr>
<td>grandFather(tom,jim).</td>
<td>father(tom,marie).</td>
<td>father(?father,?person).</td>
</tr>
<tr>
<td>grandFather(tom,ellen).</td>
<td>father(peter,bob).</td>
<td>grandFather(?grandfather,?person) if father(?grandfather,?father),</td>
</tr>
<tr>
<td>grandFather(tom,bart).</td>
<td>father(peter,jim).</td>
<td>mother(?mother,?person).</td>
</tr>
<tr>
<td></td>
<td>mother(marie,ellen).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mother(marie,bart).</td>
<td></td>
</tr>
</tbody>
</table>
Software Views with LMP

- **Extensional**
  - Logic facts (enumeration)
    ```
    class(ScExpression).
    class(ScConsExpression).
    methodInClass(analyse, ScConsExpression).
    ...
    ```

- **Intentional**
  - Logic rules (program)
    ```
    class(?method) if
    methodInClass(analyse, ?method).
    ...
    ```

Induction algorithm
Proof of concept experiment

- ScExpression
  - newClosure
  - analyse

- ScSequenceExpression
  - newClosure
  - analyse

- ScConsExpression
  - newClosure
  - analyse

- SpecialFormHandler
  - newClosure
  - handle:

- DefineHandler
  - newClosure
  - handle:

- IfHandler
  - newClosure
  - handle:
Classified Items

analyser(classImplementsMethodNamed(ScExpression,analyse)).
analyser(classImplementsMethodNamed(ScConsExpression,analyse)).
analyser(classImplementsMethodNamed(ScSequenceExpression,analyse)).
...
analyser(classImplementsMethodNamed(SpecialFormHandler,handle:)).
analyser(classImplementsMethodNamed(DefineHandler,handle:)).
analyser(classImplementsMethodNamed(IfHandler,handle:)).
Derived Rules

intention(analyser,<?class,?selector>) if
analyser(classImplementsMethodNamed(?class,?selector)).

defines intention in terms of derived rules

analyser(classImplementsMethodNamed(?class, handle:)) if
methodSendsMessage(?class, handle:, newConverterFor:),
methodSendsMessage(?class, handle:, newClosure),
methodSendsMessage(?class, handle:, analyse),
classInHierarchyOf(?class,Scheme.SpecialFormHandler),
classInHierarchyOf(?class,Scheme.SpecialFormHandlerWithSuccessor),
classInHierarchyOf(?class, ?class).

Redundant
Derived Rules

analyser(classImplementsMethodNamed(?class, analyse)) if
methodSendsMessage(?class, analyse, newClosure),
classInHierarchyOf(?class, Scheme.ScExpression),
Redundant

classInHierarchyOf(?class, ?class).

analyser(classImplementsMethodNamed(Scheme.DefineRelHandler, handle:)).
Discussion

- Results show that intentions are discovered
- Problems encountered
  - algorithm is sensitive to order of examples presented
  - sufficient number of examples is needed
    - rules are either too restrictive or too general
  - performance issues
- Scalability
  - only two small experiments, no large-scale study yet
Prototype Tool Support

Star Browser on: SchemeExpressions (4)

General Services Help

Root (1)

- SchemeExpressions (4)
  - ScSequenceExpression
  - ScConsExpression
  - ScIdentifierExpression
  - ScExtraArgumentExpression

Induce

Subclasses
Category
Protocols
Superclass
Overridden methods
Variables Used
Statements
Message sends
Class understands selectors
Arguments
Method calls
Variable Assignments
Method name
Instance Variables

Namespace
Class hierarchy

add
remove
Prolog
Induce
Extra
Inspect

<SchemeExpressions(?Var242),<hierarchy(?Var242,?Var242),hierarchy([Scheme,ScExpression],?Var242),
classInNamespace(?Var242,[Scheme]),class(?Var242)
Conclusion

- Induced intentional views combine advantages of extensional and intentional views, while removing their respective disadvantages
- Can be used to tackle software documentation problems
  - explicit link between source code and documentation by means of LMP
  - robust w.r.t. evolution
- Can be integrated easily into already existing development tools