Combining Multiple Dimensions of Knowledge in API Migration

Thiago Bartolomei, Mahdi Derakhshanmanesh, Andreas Fuhr, Peter Koch, Mathias Konrath, Ralf Lämmel, Heiko Winnebeck
Contribution of this presentation

Present a framework combining multiple dimensions of knowledge to support API migration.
API Migration

• Special case of software migration
• Adapt software system to
  – replace old API (source API) by
  – new API (target API)
  – of the same domain (e.g., GUI development)
Why API Migration?

- **Legacy** APIs may be
  - outdated, not supported anymore
- **New APIs** may provide
  - new features
  - better performance, reliability, usability, …
  - support for new system environments
Why API Migration: Example

SwingWT

- Style

Swing

SWT
Why API Migration: Example

- Compatibility

Swing

SWTSwing

SWT
API Migration Approaches
API Migration: Reimplementation
API Migration: Transformation
API Migration: Wrapping

Application \(\rightarrow\) Source API

uses

adjust

Wrapper

implements

wraps

Target API
Roadmap

• Integrated **Repository**
  – Multiple dimensions of knowledge
• **Wrapper Assessment**
  – How to rate wrappers on their suitability?
• **Guidance for Migration**
  – How to extend/write own wrappers/transformations?
INTEGRATED REPOSITORY
Metamodel

• Simplified Java sources
• API Usage properties of 1476 SourceForge projects
• Ontology on API concepts
Repository Technology: TGraphs
Repository Technology: Graph Querying with GReQL

```
from
  clsApi: V{Class}
with
  clsApi.qualifiedName =~ "javax\swing\..*" and count(clsApi-->CorrespondsTo) > 0
reportSet
  clsApi
end
```
WRAPPER ASSESSMENT
Wrapper Assessment:

Goals

• **Compare** different wrappers for same wrapping task

• **Track development** of own wrappers
Wrapper Assessment: Source API Coverage

**SwingWT**

- Packages: 7 implemented, 25 unimplemented
- Classes: 2331 implemented, 533 unimplemented
- Methods: 12506 implemented, 4533 unimplemented

**SWT Swing**

- Packages: 15 implemented, 372 unimplemented
- Classes: 281 implemented, 3426 unimplemented
- Methods: 4618 implemented, 3426 unimplemented
Wrapper Assessment: Wrapper Compliance

• Simple coverage statistics do not cover more complex dependencies
  – Declarations in supertypes
  – Empty implementations

• Simple coverage does not reflect usage of APIs in real projects
Compliance: Declarations on supertypes

SwingWT

Missing methods
- Class missing
- Class present
- Impl. in supertype

74% 22% 4%

SWT Swing

Missing methods
- Class missing
- Class present
- Impl. in supertype

78% 19% 3%
Compliance: Empty methods

SwingWT

Implemented Methods
- Impl. methods: 78%
- Empty methods: 22%

SWT Swing

Implemented Methods
- Impl. methods: 93%
- Empty methods: 7%
Wrapper Assessment: Relevance in Terms of Usage

**SwingWT**

- **Absolute**
  - Implemented: 26.6%
  - Unimplemented: 73.4%

- **Usage**
  - Implemented: 97.1%
  - Unimplemented: 2.9%

**SWTSwing**

- **Absolute**
  - Implemented: 42.6%
  - Unimplemented: 57.4%

- **Usage**
  - Implemented: 97.7%
  - Unimplemented: 2.4%
GUIDANCE FOR MIGRATION
Guidance for Migration:
Goals

- Extend existing wrappers
- Write own wrappers / reimplementations / transformations

→ Identify target API code suited to implement source API methods
Guidance for Migration: Concept-based Method Candidates
Guidance for Migration: Assessment of the Ontology

**SwingWT**

- Unimpl. meth. with link: 10.8%
- Unimpl. meth. without link: 89.2%
- Impl. meth. with link: 28.1%
- Impl. meth. without link: 71.9%

**SWT-Swing**

- Unimpl. meth. with link: 0.3%
- Unimpl. meth. without link: 99.7%
- Impl. meth. with link: 25.0%
- Impl. meth. without link: 75.0%
Guidance for Migration: Ontology Correctness

SwingWT

Methods with Links

- Correct link: 43%
- Wrong link: 57%

SWTSwing

Methods with Links

- Correct link: 37%
- Wrong link: 63%
THANK YOU FOR LISTENING!

Integrated Repository

QUESTIONS?