The Role of Context in Exception-Driven Rework

Xiang Zhao\textsuperscript{1}, Barbara Staudt Lerner\textsuperscript{2}, and Leon Osterweil\textsuperscript{1}

Laboratory for Advanced Software Engineering Research
\textsuperscript{1}University of Massachusetts Amherst
\textsuperscript{2}Mount Holyoke College

5th International Workshop on Exception Handling (WEH.12)
Zurich, Switzerland

June 16, 2012
Rework is quite common in software development processes

- Inconsistencies between requirement and design specifications cause reconsideration of both
- Inconsistencies between code and design too
- Most software engineering books ignore the topic

Rework is hard, can become very complex

- People could use help with it

Rework can be triggered by exceptions

- Addressed in an earlier paper of ours
Exception-Driven Rework Pattern

- Previously proposed exception handling pattern

\[ \text{Work} \rightarrow \text{Check Work} \rightarrow \text{Rework} \]

Agent: fixer
Agent: anomaly detector

Exception instances as triggers
Context support

\[ \text{Work} \rightarrow \text{Check Work} \rightarrow \text{Rework} \]

\[ \text{Agent: fixer} \]

\[ \text{Agent: anomaly detector} \]

Repeat activities that had been done previously in an earlier phase triggered by exception instance(s)
Reconsideration of revision on previously performed activities
Reinstantiation of earlier steps in new contexts

\[ \text{Xiang Zhao}^1, \text{Barbara Staudt Lerner}^2, \text{and Leon Osterweil}^1 \]

The Role of Context in Exception-Driven Rework

\[ ^1 \text{Lerner et al. TSE Vol.36, 2010} \]
Exception-Driven Rework Pattern

- Previously proposed exception handling pattern\(^2\)

- Repeating activities that had been done previously in an earlier phase triggered by exception instance(s)

\(^2\)Lerner et al. TSE Vol.36, 2010

Xiang Zhao\(^1\), Barbara Staudt Lerner\(^2\), and Leon Osterweil\(^1\)

The Role of Context in Exception-Driven Rework
Exception-Driven Rework Pattern

- Previously proposed exception handling pattern\(^2\)

- Repeating activities that had been done previously in an earlier phase triggered by exception instance(s)
  - Reconsideration and revision of previously performed activities
  - Reinstantiation of earlier steps in new contexts

\(^2\)Lerner et al. TSE Vol.36, 2010
Refactoring is rework of design
  - May or may not be triggered when code is recognized as being untidy
  - There are many different design patterns [Fowler 1999]
Refactoring as an Example of Exception-Driven Rework

- Refactoring is rework of design
  - May or may not be triggered when code is recognized as being untidy
  - There are many different design patterns [Fowler 1999]
- Separate Query from Modifier Refactoring
  - Splits a method that was both a query and a modifier into two methods
    - Create a query method to return the same value
    - Change the return statement in original method to return the query
    - Add calls to the query before the calls to the original method
    - Change the original method to void and remove its return statements
Refactoring as an Example of Exception-Driven Rework

- Refactoring is rework of design
  - May or may not be triggered when code is recognized as being untidy
  - There are many different design patterns [Fowler 1999]
- Separate Query from Modifier Refactoring
  - Splits a method that was both a query and a modifier into two methods
    - Create a query method to return the same value
    - Change the return statement in original method to return the query
    - Add calls to the query before the calls to the original method
    - Change the original method to void and remove its return statements
- Executing this rework process can entail carrying out a number of different kinds of rework
Exception instances are handled differently according to their types.

Each exception instance triggers rework.
Exception instances are handled differently according to their types.

Each exception instance triggers rework.
Exception-Triggered Rework Examples

Change return statement
Create query method
Compile
Run unit tests
Handle Unit Test Compilation Failure
Change return statement
Create query method
Compile
Run unit tests
Handle Unit Tests Failure
Handle Unit Tests Failure
UnitTestCompilationFailureException
UnitTestFailureException

Rework is modeled very accurately as recursive step invocations. Actual rework should be guided by context provided by artifact values.

Xiang Zhao¹, Barbara Staudt Lerner², and Leon Osterweil¹
The Role of Context in Exception-Driven Rework
Rework is modeled very accurately as recursive step invocations.
Rework is modeled very accurately as recursive step invocations.

Actual rework should be guided by context provided by artifact values.
Typical Questions that Users Want Answers to during Rework

- Where am I?
- What am I doing here?
- How did I get here?
- How did that work out?
- What alternatives do I have now?
- Which are likely to turn out best?
Typical Questions that Users Want Answers to during Rework

- Where am I?
- What am I doing here?
- How did I get here?
- How did that work out?
- What alternatives do I have now?
- Which are likely to turn out best?

Contextual information could help
Context Information that could Help

- Present process execution state
  - Current artifact values
  - Pointers to executing steps and their recursions
- A complete process execution history
  - Prior values of artifacts
  - Previous step execution sequences
- Information that could help to form a plan for getting back to “normal”
- etc.
Proposed Support for Exception-Driven Rework

- Rigorous, executable process definition
  - Can provide articulate artifact flows and control flows.
  - Can use scoping to assure visibility of only appropriate artifacts
- Process Introspection
  - The ability to examine current process states
- Process Retrospection
  - Ability to examine past process execution history
- Data Derivation Graph provides the previous two
Defined templates for translating Little-JIL step executions into DAG fragments
Defined templates for translating Little-JIL step executions into DAG fragments

Basic Features
- Represents how artifacts are derived from each other
- Incorporates scoping, nesting, hierarchy information

Additional Features
- Links to previous artifacts values
- Detailed history is inferable
- Can generate DDGs dynamically while the process is executing

Xiang Zhao¹, Barbara Staudt Lerner², and Leon Osterweil¹

The Role of Context in Exception-Driven Rework
Defined templates for translating Little-JIL step executions into DAG fragments

Basic Features
- Represents how artifacts are derived from each other
- Incorporates scoping, nesting, hierarchy information

Additional Features
- Links to previous artifacts values
- Detailed history is inferable
• Defined templates for translating Little-JIL step executions into DAG fragments

• Basic Features
  • Represents how artifacts are derived from each other
  • Incorporates scoping, nesting, hierarchy information

• Additional Features
  • Links to previous artifacts values
  • Detailed history is inferable

• Can generate DDGs dynamically while the process is executing
How is it Supposed to be Helpful?

Q: What did I do to (the same part of) the source code when I was trying to fix an issue caused by test case failure, which may possibly be the reason why the compilation fails right now?
Q: What did I do to (the same part of) the source code when I was trying to fix an issue caused by test case failure, which may possibly be the reason why the compilation fails right now?

A: See the DDG
Q: What did I do to (the same part of) the source code when I was trying to fix an issue caused by test case failure, which may possibly be the reason why the compilation fails right now?
How is it Supposed to be Helpful?

DDG suffers from scalability issues!

Xiang Zhao¹, Barbara Staudt Lerner², and Leon Osterweil¹

The Role of Context in Exception-Driven Rework
Future Work

- A DDG Query Language
  - How to support asking questions during rework?
  - How to provide clear answers quickly?

Ripple effects

Support for helping users decide the order in which to handle exceptions when many are thrown

Probably can use prospection for this

Study more refactoring patterns

Xiang Zhao¹, Barbara Staudt Lerner², and Leon Osterweil¹

The Role of Context in Exception-Driven Rework
Future Work

- A DDG Query Language
  - How to support asking questions during rework?
  - How to provide clear answers quickly?
- Ripple effects
  - Support for helping users decide the order in which to handle exceptions when many are thrown
    - Probably can use prospection for this
- Study more refactoring patterns
Related Work

- Exception Handling in Workflow Systems
  - [Eder et al. EDBT ’98] discussed the concept of handling exceptions with partial rollback and forward execution.
  - Event driven architectures (EDA)

- Rework Formalization
  - [Cass et al. EWSPT] proposed initial approaches of formalizing rework.
  - A pattern for modeling rework [Cass et al. ICSP ’09]

- Context Support
  - [Antunes et al. AITSE ’10] proposed a context model in software development with multiple layers and perspectives.
  - Mylyn [12] is a tool integrating task management and task context [Kersten et al. AOSD ’05]
Thank You

- Questions?