Supporting Process Undo and Redo in Software Engineering Decision Making

Xiang Zhao, Yuriy Brun, Leon J. Osterweil

University of Massachusetts Amherst
{xiang,brun,ljo}@cs.umass.edu
Creative processes often require reworking decisions [ICSSP ’12]

Goal: Provide automated support for

- Undo: Go to an earlier state of development
- Revisit: Modify an earlier decision to correct the mistake
- Redo: Save work by automatically reapplying previously completed tasks to modified artifacts

Example

Planning a trip: a simple scenario that may involve undo/redo
A group of people plan a trip

- pick a date for the trip
- pick a city as the destination
- Everyone reserve a flight
Planning a Trip

- A group of people plan a trip
  - **pick a date** for the trip
  - **pick a city** as the destination
  - Everyone **reserve a flight**

Data Derivation Graph (DDG) stores the process provenance:
- data creations, usage, and modifications
- step execution sequences
- scoping
Planning a Trip

- A group of people plan a trip
  - pick a date for the trip
  - pick a city as the destination
  - Everyone reserve a flight

Data Derivation Graph (DDG) stores the process provenance:
- data creations, usage, and modifications
- step execution sequences
- scoping
Planning a Trip

- A group of people plan a trip
  - **pick a date** for the trip
  - **pick a city** as the destination
  - Everyone **reserve a flight**

Data Derivation Graph (DDG) stores the process provenance:
- data creations, usage, and modifications
- step execution sequences
- scoping
A group of people plan a trip
- **pick a date** for the trip
- **pick a city** as the destination
- Everyone **reserve a flight**

Data Derivation Graph (DDG) stores the process provenance:
- data creations, usage, and modifications
- step execution sequences
- scoping
A group of people plan a trip
- pick a date for the trip
- pick a city as the destination
- Everyone reserve a flight

reserve a flight requires going back to pick a date

Data Derivation Graph (DDG) stores the process provenance:
- data creations, usage, and modifications
- step execution sequences
- scoping
Going Back to pick a date Step

Our approach

- **Undo**
  - tasks while remembering old artifacts and consequences
  - undo step brings the process state back to pick a date

- **Revisit**
  - an old task while storing the provenance of undone tasks
  - pick a date step is revisited for another decision

- **Redo**
  - set-aside tasks that are consistent with the modification
  - pick a city and reserve a flight steps are automatically redone

What drives the undo/redo process?

```
java
public class UndoPickDateException {
    public static void main(String[] args) {
        // pick a date
        state
        // pick a city
        state
        // reserve a flight
        state
    }
}
```
Going Back to pick a date Step

Our approach

- undo step brings the process state back to pick a date
- undo step is revisited for another decision
- redo set-aside tasks that are consistent with the modification

What drives the undo/redo process?

```
state
state
date= 5/17
city=
flight=
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/17
city=
flight=
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=
flight=
```

```
state
state
date= 5/18
city=
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
flight=
```

```
state
state
date= 5/18
city=SF
```
Our approach

- **Undo** tasks while remembering old artifacts and consequences

  - **undo** step brings the process state back to **pick a date**
Going Back to pick a date Step

Our approach

- **Undo** tasks while remembering old artifacts and consequences
  - *undo* step brings the process state back to *pick a date*
- **Revisit** an old task while storing the provenance of undone tasks
  - *pick a date* step is revisited for another decision
Going Back to pick a date Step

Our approach

- **Undo** tasks while remembering old artifacts and consequences
  - undo step brings the process state back to pick a date
- **Revisit** an old task while storing the provenance of undone tasks
  - pick a date step is revisited for another decision
- **Redo** set-aside tasks that are consistent with the modification
  - pick a city and reserve a flight steps are automatically redone

What drives the undo/redo process?

- **UndoPickDateException**

![Diagram](image-url)
Going Back to pick a date Step

Our approach

- **Undo** tasks while remembering old artifacts and consequences
  - *undo* step brings the process state back to **pick a date**
- **Revisit** an old task while storing the provenance of undone tasks
  - **pick a date** step is revisited for another decision
- **Redo** set-aside tasks that are consistent with the modification
  - **pick a city** and **reserve a flight** steps are automatically redone

What drives the undo/redo process?
Process Support for Undo/Redo

Plan a trip

- Pick a date
- Pick a city
- Reserve a flight

Exception handlers can be recursive to assist repetitive undo/redo.
Process Support for Undo/Redo

- Plan a trip
- Pick a date
- Pick a city
- Reserve a flight
- Undo pick a date
- Undo
- Undo pick a date
- Plan a trip
- +

Exception handlers can be recursive to assist repetitive undo/redo.
**undo pick a date** handles the exception requesting undoing **pick a date**
Process Support for Undo/Redo

- **undo pick a date** handles the exception requesting undoing **pick a date**
- **undo** retrieves the execution state vector at a selected point.
**Process Support for Undo/Redo**

- **undo pick a date** handles the exception requesting undoing pick a date
- **undo** retrieves the execution state vector at a selected point.
- **plan a trip** is redone, led by pick a date
undo pick a date handles the exception requesting undoing pick a date
undo retrieves the execution state vector at a selected point.
plan a trip is redone, led by pick a date
Exception handlers can be recursive to assist repetitive undo/redo.
We define a specific exception handler for each kind of undo.
We built a system that
1. automatically records detailed process execution history
2. extracts process state at any given point
3. overrides the current state with the retrieved state, and drives the process forward

supporting
1. **Undoing** tasks while remembering old artifacts and consequences
2. **Revisiting** an old task while storing the provenance of undone tasks
3. **Redoing** set-aside tasks that are consistent with the modification automatically

A more complex (and practical) example

**Tease Apart Inheritance Refactoring**
Fix an inheritance hierarchy that is doing two jobs at once
Fix an inheritance hierarchy that is doing two jobs at once

Dimensions
Type of the NewsFeedItem
Source of the NewsFeedItem
Tease Apart Inheritance Refactoring (TAIR) [Fowler 1999]

Fix an inheritance hierarchy that is doing two jobs at once

Dimensions
Type of the NewsFeedItem
Source of the NewsFeedItem

Code Duplication
FacebookPost.authorLink()
FacebookLink.authorLink()
Decision Making in TAIR Process

Which dimension to extract?
- Source
- Type

Which fields to move?
- content
- author

Which methods to move?
- content()
- authorLink()

Other issues
- construction method?
- inherited field?
- inherited method?

Need to keep code consistency in the whole process!
Decision Making in TAIR Process

Which dimension to extract?
- Source
- Type

Which fields to move?
- content
- author

Which methods to move?
- content()
- authorLink()

Other issues
- construction method?
- inherited field?
- inherited method?

Need to keep code consistency in the whole process!
Which dimension to extract?
- Source
- Type

Which fields to move?
- content
- author

Other issues
- construction method?
- inherited field?
- inherited method?

Need to keep code consistency in the whole process!
Which dimension to extract?
- Source
- Type

Which fields to move?
- content
- author

Which methods to move?
- content()
- authorLink()
Decision Making in TAIR Process

Which dimension to extract?
- Source
- Type

Which fields to move?
- content
- author

Which methods to move?
- content()
- authorLink()

Other issues
- construction method?
- inherited field?
- inherited method?

Need to keep code consistency in the whole process!
Undo and Redo in TAIR Decision Making
Undo and Redo in TAIR Decision Making
Undo and Redo in TAIR Decision Making

Type is being extracted, and after a few steps...
Undo and Redo in TAIR Decision Making

Need to fix fields, methods, constructors to maintain the inheritance structure. Facebook and Twitter should not be subclasses of NewsFeedItem. It might be the dimension we chose to extract were wrong. How to fix it? We can undo a series of steps, revert the artifacts to an earlier state in the history, and redo the necessary steps to get the desired state.
Undo and Redo in TAIR Decision Making

Need to fix fields, methods, constructors to maintain the inheritance structure.

Facebook and Twitter should not be subclasses of NewsFeedItem.

The field we tried to move was wrong?
The dimension we chose to extract was wrong?
Undo and Redo in TAIR Decision Making

Need to fix fields, methods, constructors to maintain the inheritance structure.

Facebook and Twitter should not be subclasses of NewsFeedItem

The field we tried to move was wrong?
The dimension we chose to extract was wrong?

How to fix it?
We can **undo** a series of steps, bring the artifacts to an earlier state in the history, and **redo** the necessary steps to get the desired state.
Using the DDG to Undo/Redo

**Scenario**
The developer decides to choose a different field to move to the new hierarchy.
Using the DDG to Undo/Redo

---

**Scenario**

The developer decides to choose a different field to move to the new hierarchy

- *author* field is selected to move

---

*state*

dimension = Type

*sourcecode*

fieldname = null

---

*state*

dimension = Type

*sourcecode*

fieldname = author

---

Scenario

The developer decides to choose a different field to move to the new hierarchy

- *author* field is selected to move
Scenario
The developer decides to choose a different field to move to the new hierarchy

- author field is selected to move
- Too many inconsistencies are spotted while updating references: Throw an exception for undoing Pick a field [0]
Using the DDG to Undo/Redo

**Scenario**

The developer decides to choose a different field to move to the new hierarchy

- *author* field is selected to move
- Too many inconsistencies are spotted while updating references: Throw an exception for undoing **Pick a field[0]**
- **Undo**: Process state is brought back to **Pick a field[0]**
Using the DDG to Undo/Redo

Scenario
The developer decides to choose a different field to move to the new hierarchy

- *author* field is selected to move
- Too many inconsistencies are spotted while updating references: Throw an exception for undoing *Pick a field[0]*
- **Undo**: Process state is brought back to *Pick a field[0]*
Using the DDG to Undo/Redo

Scenario
The developer decides to choose a different field to move to the new hierarchy

- **author** field is selected to move
- Too many inconsistencies are spotted while updating references: Throw an exception for undoing **Pick a field[0]**
- **Undo**: Process state is brought back to **Pick a field[0]**
- **Revisit**: Set `fieldname = content`
Scenario
The developer decides to choose a different field to move to the new hierarchy

- **author** field is selected to move
- Too many inconsistencies are spotted while updating references: Throw an exception for undoing **Pick a field [0]**

- **Undo**: Process state is brought back to **Pick a field [0]**
- **Revisit**: Set **fieldname = content**
- **Redo**: Move **content**
Using the DDG to Undo/Redo

Scenario
The developer decides to choose a different field to move to the new hierarchy

- *author* field is selected to move
- Too many inconsistencies are spotted while updating references: Throw an exception for undoing *Pick a field[0]*

**Undo**: Process state is brought back to *Pick a field[0]*

**Revisit**: Set `fieldname = content`

**Redo**: Move `content`

Redo may indicate an error in an earlier decision about extracting `Type`
Using the DDG to Undo/Redo

- Pick a field [0]
- Remove Field in Source
- Update References
- Undo [0]

state: sourcecode
dimension = Type
tfieldname = null

- Pick a field [1]
- Remove Field in Source

state: sourcecode
dimension = Type
tfieldname = content
Using the DDG to Undo/Redo

- **Pick a field [0]**
- **Remove Field in Source**
- **Update References**
- **Undo [0]**
- **UndoMoveFieldException**

- **Pick a field [1]**
- **Remove Field in Source**
- **Undo [1]**

- **Create Dimension Class**
  - **sourcecode**
  - **dimension = Type**
  - **fieldname = null**

- **Extract Dimension**
  - **sourcecode**
  - **dimension = null**
  - **fieldname = null**

- **state**
  - **dimension = Type**
  - **fieldname = null**

- **state**
  - **dimension = Type**
  - **fieldname = author**

- **state**
  - **dimension = Type**
  - **fieldname = author**

- **state**
  - **dimension = Type**
  - **fieldname = content**

- **state**
  - **dimension = null**
  - **fieldname = null**

- **state**
  - **dimension = Source**
  - **fieldname = null**
Using the DDG to Undo/Redo

Pick a field [0]
Remove Field in Source
Update References
Undo [0]

Pick a field [1]
Remove Field in Source

Create Dimension Class

state
dimension = Type
fieldname = null

state
dimension = Type
fieldname = author

state
dimension = Type
fieldname = author

state
dimension = Type
fieldname = null

state
dimension = Source
fieldname = null

UndomoveFieldException

state
dimension = Type
fieldname = null

state
dimension = null
fieldname = null

state
dimension = Source
fieldname = null
We define a complete executable process definition for TAIR
Related Work

- **Undo Mechanism**
  - [Leeman TPLS ’86] proposed a formal approach to undo operations.
  - Selective undo model [Berlage TCHI ’94] provides the user with the ability to undo an arbitrary operation in history.
  - Script undo model [Archer et al. TPLS ’84] treats the undo operations as the editing of a script of commands.
  - *Our approach takes into account both control flow and data state.*

- **Provenance Visualization**
  - Provenance Map Orbiter [Seltzer et al. TaPP ’11] captures *large* provenance graphs and provides navigation mechanism.
  - Navigation model for scientific provenance [Anand et al. WORKS ’09].
  - *DDG takes advantage of Little-JIL’s hierarchical structure.*

- **Refactoring**
  - Refactoring is widely supported in modern IDEs, few of which care about contexts.
  - Mylyn [Kersten et al. AOSD ’05] emphasizes contexts.
  - *A rigorous process model is needed to understand the artifacts.*
Contributions and Future Work

Contributions:
- Undo tasks
- Revisit an old task
- Automatically redo set-aside tasks
- An eclipse plug-in

Future Work:
- User interface for browsing and querying the DDG
- Detect conflicts in redo operations
- Add flexibility to Little-JIL