Towards Assessing the Technical Debt of Undesired Software Behaviors in Design Patterns

Derek Reimanis
Clemente Izurieta
Current TD Analysis Tools

- SonarQube (TD plugin)
  - SQALE model
- CAST
  - Prioritization of violations
- inFusion
  - Design disharmonies

Structural

Behavioral
Position

• More effort needs to be directed at understanding software behaviors as they impact TD
• Why?
  • Further the research area
  • Expand TD analysis tools’ capabilities
Software Behavior

• How do we talk about behavior?
• Internal
  • Interior mechanisms and API calls
• External
  • Observable results
Design Pattern Characteristics

- Known best practice
- Formalizations
  - SPS and IPS
- Grime (structural)
  - Unintended artifacts that muddy a pattern instance
Observer Pattern IPS

n: /Notify()

/s:[[Subject]]

n.p * : /Update(s)

n.p.q : st := /GetState()

: [[Observer]]

1.1: NotifyTemp()

/s:Kiln

1.1.1 * : Update(s)

1.1.1.1 : LogUpdateRecd(s)

1.1.1.2 : st := GetKilnTemp()

: TempObs

Excessive Action(s)

• Excessive action(s)
  • Action(s) that are un-essential to the functional runtime behavior of the pattern
  • Same external behavior, different internal behavior

• Created by:
  • Forgetting to remove debugging code
  • Changing implementation details
Excessive Action(s)

```java
public class RemoteTempObserver1 extends AbstractRemoteTempObserver {

    @Override
    protected void update(AbstractKiln kilnToUpdate) {
        // excessive behavior
        for (int j = 0; j < 100; j++) {
            j++; // error: j++ instead of j++;
        }
        this.displayedTemp = kilnToUpdate.getTemperature();
    }
}
```

1.1: NotifyTemp()

1.1.1 * : Update(s)

1.1.1.1 : Excessive()

/s: Kiln

1.1.1.2 : st := GetKilnTemp()
Improper Order of Sequences

• Improper Order of Sequences
  • Incorrect order of method/class calls during runtime execution
  • Same external behavior, different internal behavior

• Created by:
  • Unfamiliarity with design pattern
public class RemoteTempObserver1 extends AbstractRemoteTempObserver {

    @Override
    protected void update(AbstractKiln kilnToUpdate) {

        // improper order of sequences
        grime.update(kilnToUpdate);

        this.displayedTemp = kilnToUpdate.getTemperature();
    }
}
Improper Order of Sequences

1.1: NotifyTemp()

/s: Kiln

1.1.1 *: Update(s)

1.1.1.1: LogUpdateRecd(s)

: TempObs

1.1.1.2: st := GetKilnTemp()

1.1.1.2: Construct()

1.1.1.3: Update(s)

TempObs2
Conclusion and Future

• More effort needs to be directed at understanding software behaviors as they impact TD
• Find more errant behaviors
• Identify/proposal behavioral metrics
• Benchmarking
• Implementation in tools
Questions