Effects of Dependency Injection on Maintainability

Kate Razina

Overview

- Introduction
  - Maintainability
  - Dependency Injection
- Hypothesis
- Research
  - Measuring Maintainability
  - Data Collection
- Results
- Conclusion
  - Future Work

Maintainability

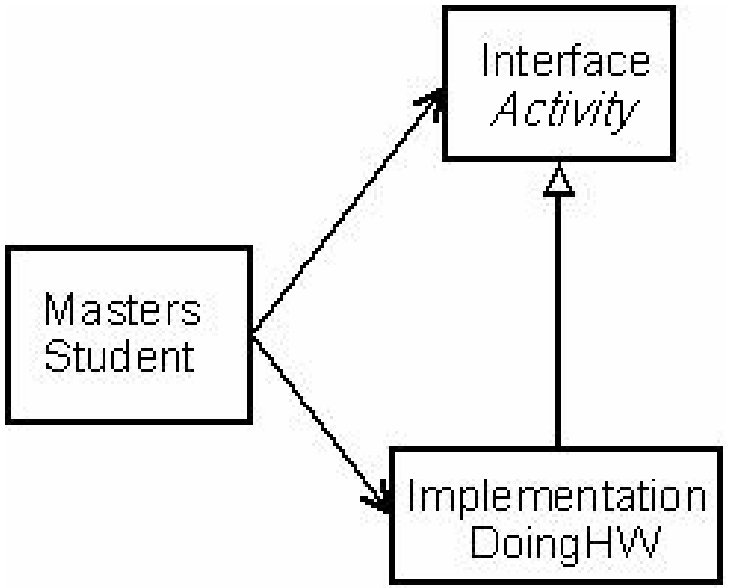
- Definition

  The ease with which a software system or component can be modified. Modifications may include extensions, porting to different computing systems or improvements[1]

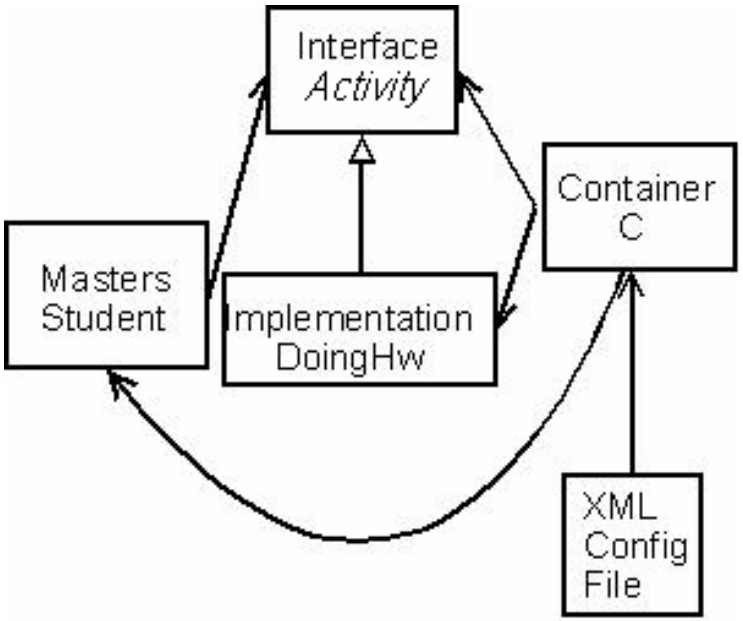
- Maintainability
  - Most time consuming part of software life cycle
  - 65% to 75% of total time[2]

1. M. Mari and N. Eila. The impact of maintainability on component-based software systems. In Euromicro Conference, pages 25–32, 2003.
2. S. Muthanna, K. Konotogiannis, K. Ponnambalam, and B. Stacey. A maintainability model for industrial software systems using design level metrics. In Seventh Working Conference on Reverse Engineering, pages 248–256, November 2000.

Dependency Injection

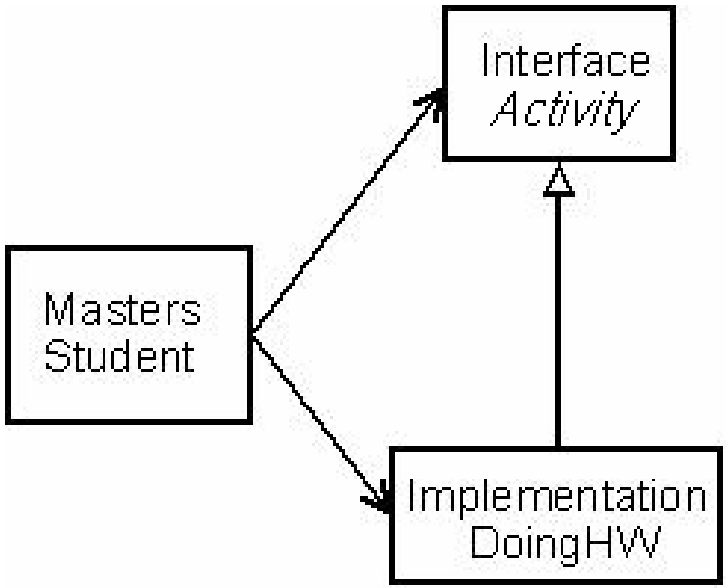


No Dependency Injection

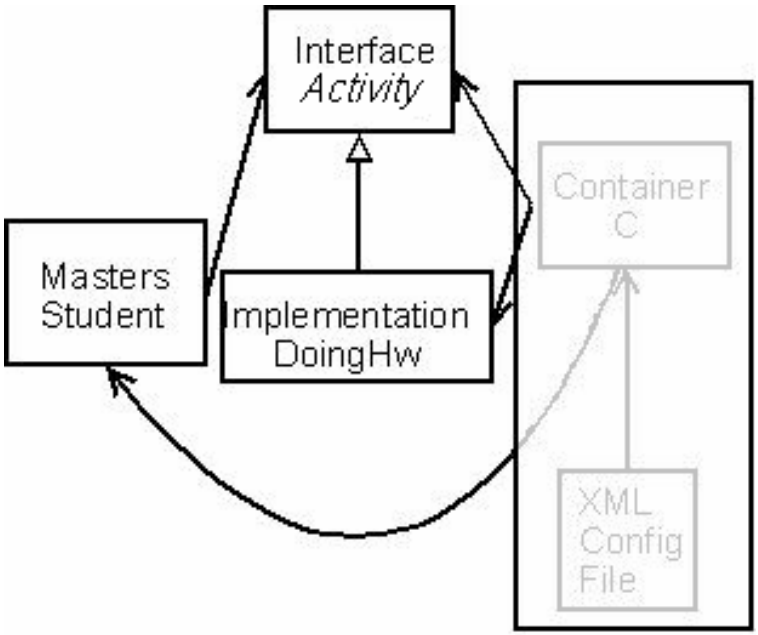


Dependency Injection

Dependency Injection



No Dependency Injection



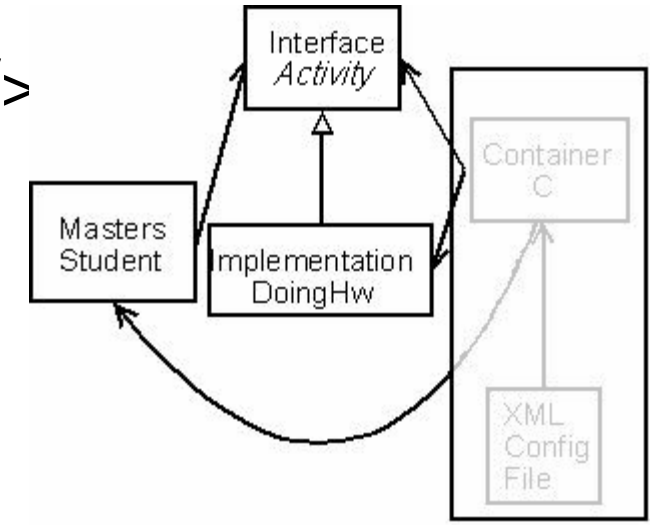
Dependency Injection

Dependency Injection (IOC)

- Types
  - Constructor
  - Setter
- Used in
  - Spring framework

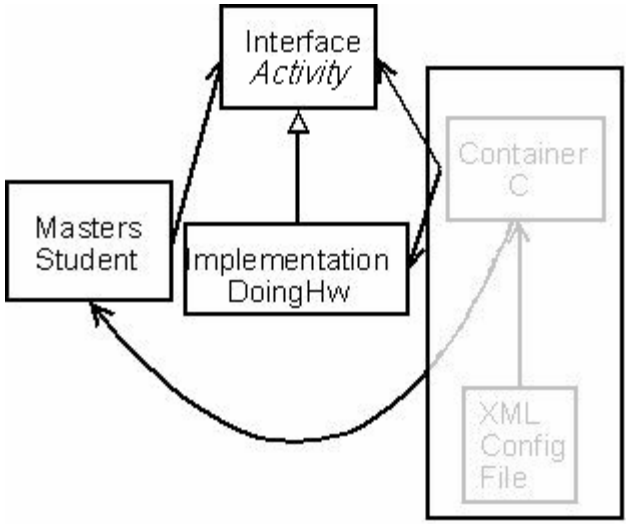
Dependency Injection XML File (Constructor Injection)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
    <bean id="activityHW" class="student.DoingHw"/>
    <bean id="student" class = student.MastersStudent">
          < constructor-arg>
                <ref bean="activityHW"/>
          </constructor-arg>
    < /bean >
<beans>
```



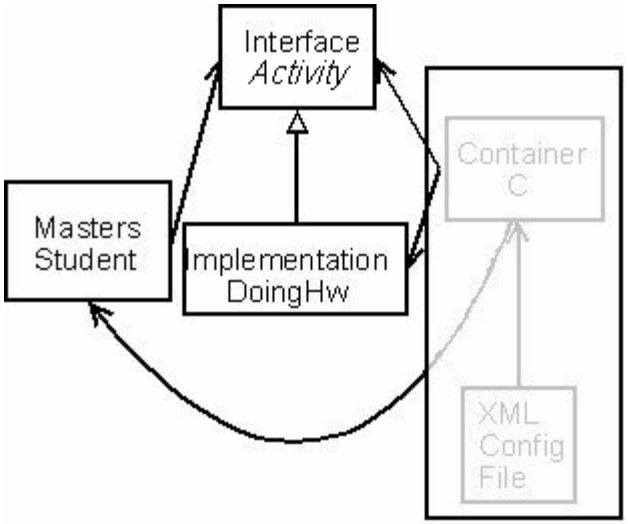
Dependency Injection Java File (Constructor Injection)

```
public class MastersStudent implements Student{
        private Activity activity;
        public MastersStudent(Activity activity){
                this.activity = activity;
        }
}
```



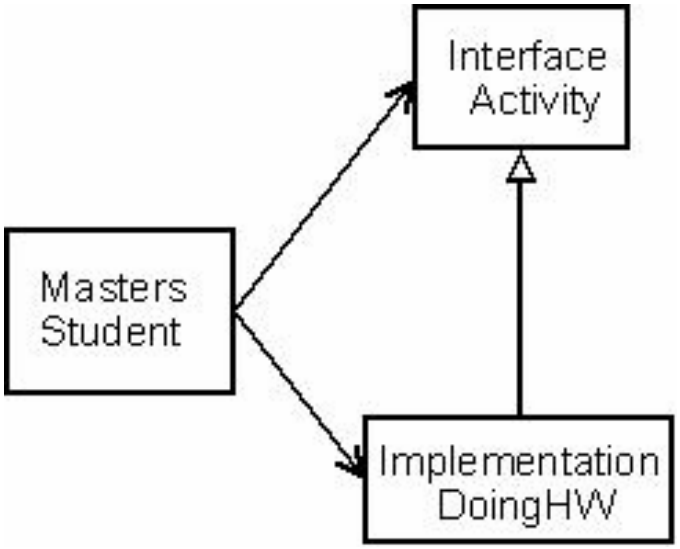
Why Use Dependency Injection?

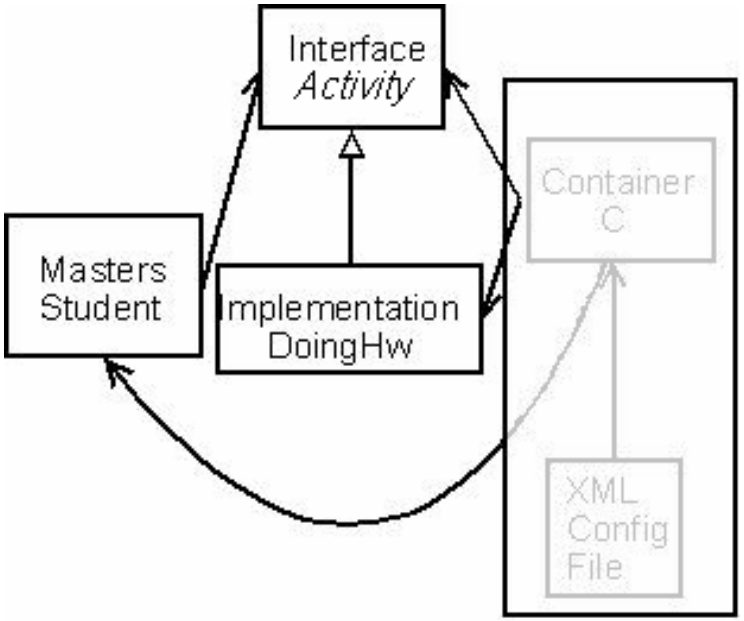
- Container is provided, configured with an XML file
- A MastersStudent can do any activity with a slight change of the XML file
- We can test MastersStudent without subsequently testing the implementation of Activity
- Responsibility for coordination collaboration between objects is transferred away from the objects themselves



Why Use Dependency Injection?

- Coupling between objects is reduced or is it?
  - Decoupled, testable, easy to maintain[3]
- Couplings become more configurable and flexible.





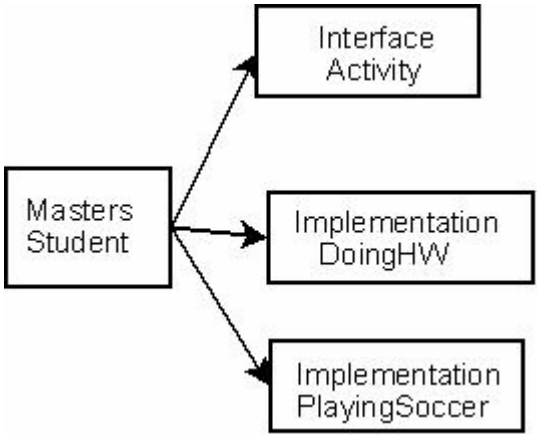
3. Walls, Craig. Spring in Action. Manning, 2005.

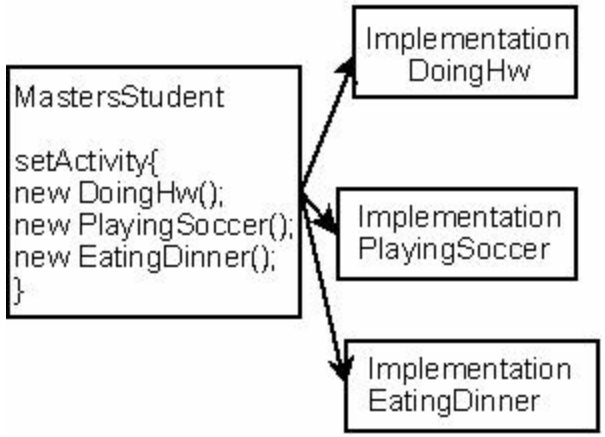
Hypothesis

- The pattern of dependency injection significantly reduces dependencies between classes in a piece of software, therefore making the software more maintainable.

How Do We Measure Maintainability?

- Single Metric Measures
  - Cohesion (LCOM)
  - Coupling
    - CBO
      - 3
    - RFC
      - 4





How Do We Measure Maintainability?

- Lower coupling
  - Improved maintainability
  - Decreases interdependencies between modules[4]
  - The higher the coupling between modules, the more difficult the modules are to understand, change, and correct [5, 6]

4. M. Dagpinar and J. H. Jahnke. Predicting maintainability with object oriented metrics -an empirical comparison. In 10th Working Conference on Reverse Engineering, 2003.
5. L. Briand, J. Daly, and J. Wust. A unified framework for cohesion measurement in object oriented systems. Empirical Software Engineering: An International Journal, 3(1):65–117, March 1998.
6. R.W. Selby and V.R. Basili. Analyzing Error-Prone Systems Structure. IEEE Trans. Software Eng. Vol. 17, no. 2, pp. 141-152.

Data Collection

- Spring Framework
- Project Collection
  - Sourceforge.net
  - 40 projects

Data Collection

- CKJM Tool
  - Metrics
    - CBO, RFC, LCOM
    - Weighted Methods per Class (WMC)
    - Depth of Inheritance Tree (DIT)
    - Number of Children (NOC)
- Number of DIs Metric
  - Measure to what extent each Spring project uses dependency injection

Data Collection

- Analysis
  - 2 Sample t-test
    - p-value <= 0.05

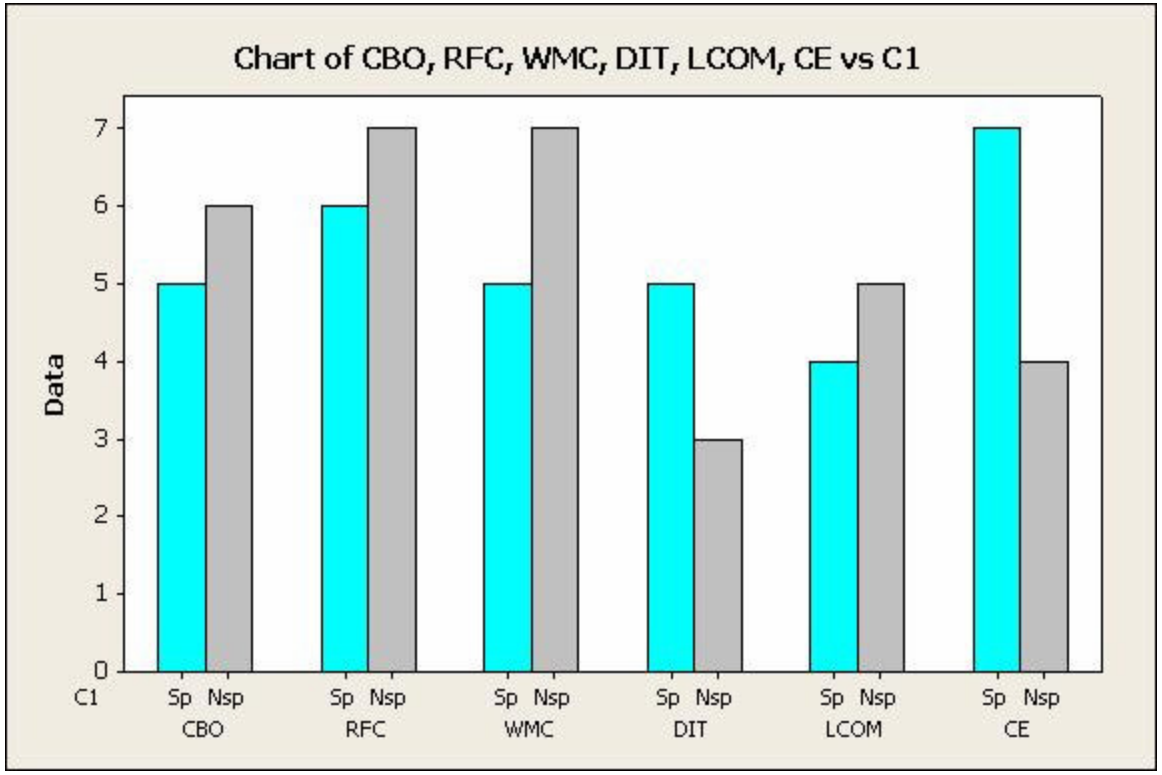
Results

- T-Test

	CBO	RFC	WMC	DIT	LCOM	CE
Spring	5	6	5	5	4	7
Non-Spring	6	7	7	3	5	4

Results

- T-Test

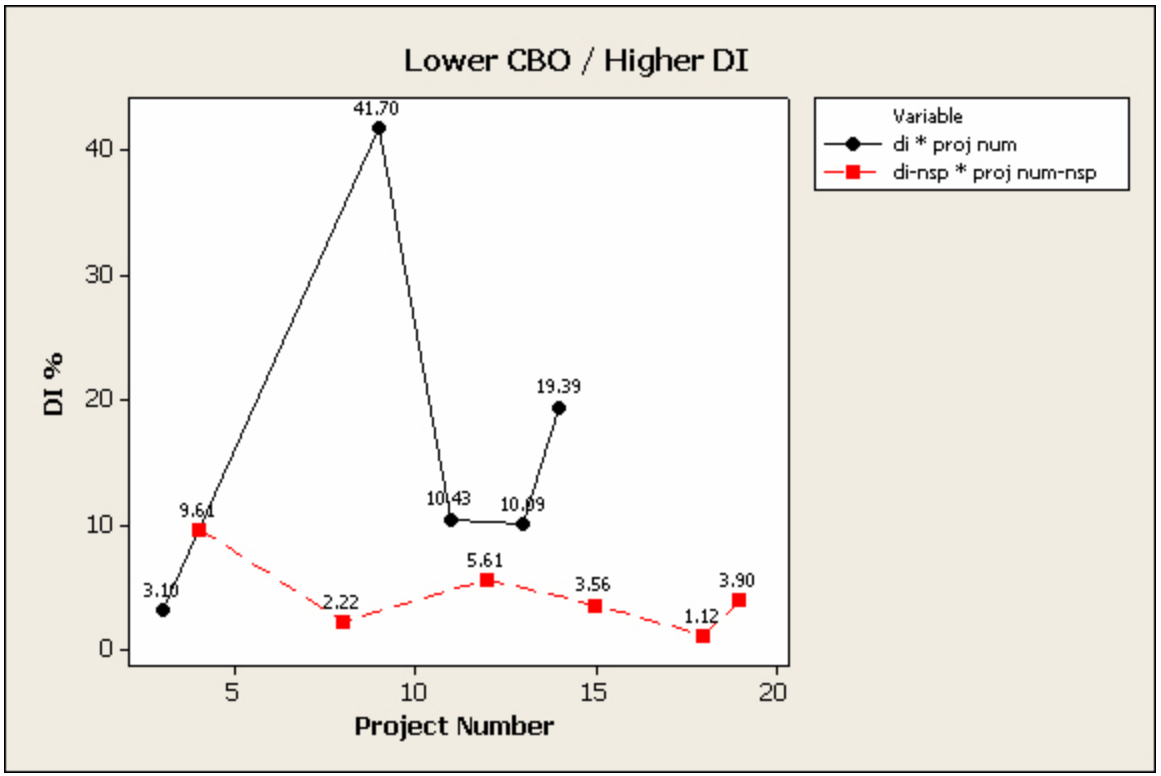


Results

- No obvious correlation exists between the presence of dependency injection and the Chidamber and Kemmerer metrics.
- Examine CBO & RFC results closely
  - Correlation between lower average CBO/higher DI percentage and vise-versa
  - Correlation between lower average RFC/higher DI percentage and visa-versa

Results

	CBO
Spring	5
Non-Spring	6



Conclusion

- Hypothesis:
  - The pattern of dependency injection significantly reduces dependencies between classes in a piece of software, therefore making the software more maintainable.
- Results:
  - No obvious correlation exists between the presence of dependency injection and average CBO and RFC
  - We cannot say that using dependency injection will lower coupling and therefore reduce maintainability
  - Correlation between lower average CBO/higher DI percentage and vise-versa
  - Correlation between lower average RFC/higher DI percentage and visa-versa

Future Work

- Study coupling more
- Ideal percentage of DI
  - Burden of maintenance now includes xml code as well as java code
- DI / ∑CBO metric

Questions?