“I pledge my honor that I have abided by the Stevens Honor System.”

Empirical software engineering Hypothesis:

- Implementing test-driven development (TDD) will lead to higher code quality and fewer defects in the software products.

Independent variable:

- Use of TDD

Dependent variable:

- Code quality
- Number of defects

Control variables:

- development team size
- experience level of the developers
- programming language used

Threats to conclusion validity:

- factors not related to TDD that may impact code quality and defects (changes in development processes or tools)

Threats to construct validity:

- Inaccurate measurement of code quality and defects

Threats to internal validity:

- Changes in the development team or other external factors that may impact code quality and defects

Threats to external validity:

- The results may not be generalizable to other development teams or projects

Data collection:

- We can collect data on code quality and defects for a period of time before and after the implementation of TDD.
- This can be done through manual code reviews, automated testing tools, and defect tracking systems.

Using the data:

- Once the data has been collected, by comparing the data collected before and after the implementation of TDD, we can determine whether there was a significant improvement in code quality and a reduction in defects.
- If the data shows a positive impact of TDD on code quality and defects, we can conclude that TDD is an effective practice and consider implementing it in other projects.
- If the data does not show a positive impact, we may need to reconsider the use of TDD and look for alternative ways to improve code quality and reduce defects.