**Introduction:**

Software maintenance represents a vital development stage which uses up most system resources throughout the entire lifecycle. Systems become less maintainable when they develop internal structural problems during evolution for user need fulfillment or bug correction. The research investigates software quality enhancement through reengineering as a methodology which transforms existing systems into higher performing models with better maintainability features and reduced expenditures. The authors use Agile Scrum methodology supported by the Chidamber and Kemerer Java Metrics (CKJM) tool for validating enhancements. Design metrics of the system undergo examination through NetBeans 7.3 after implementing the reengineering process and before system implementation.

**Reengineering Process:**

Reengineering involves three stages:

* Fundamental systems analysis through existing documentation and code evaluation performs Reverse Engineering.
* Modification of current code structures happens through alterations which builds new components according to project requirements.
* The process of forward engineering incorporates new elements before executing system tests to confirm reliability and correctness. Enhancing software quality remains the objective even though an adjustment of how the application functions externally is not required.

**Quality Metrics:**

Six quality metrics from the CK suite are utilized in the research study.

**WMC** (Weighted Methods per Class) – Indicates class complexity.

**CBO** (Coupling Between Object Classes) – Measures class dependency.

**RFC** tracks every method execution that occurs when receiving a message to a class.

**LCOM** (Lack of Cohesion of Methods) – Reflects internal class cohesiveness.

**DIT** (Depth of Inheritance Tree) – Depth from a class to the root.

**NOC** (Number of Children) – Number of immediate subclasses.

The metrics function as markers to evaluate system complexity as well as cohesion and maintainability.

**Findings and Results:**

1. **Design Quality Improvement:**

There was substantial improvement observed in all three classes namely Login along with IDE and User Detail.

The breakdown of WMC by more than 50% demonstrated that classes became simpler in design.

Both LCOM values decreased considerably resulting in improved class cohesion while evaluation progressed.

A decrease in CBO and RFC values showed the results of minimal method dependencies between classes.

No changes occurred to class hierarchies because the values of DIT and NOC remained stable.

The series of improvements benefit system maintainability while decreasing the time required for future development projects.

1. **Execution Time Reduction:**

A review of thirty-five Mean Time to Execute (MTTE) measurements demonstrates:

• Overall execution time dropped by 6.5%.

Execution time for the Login class was decreased from its initial value of 123.2ms to 106.4ms.

The User Detail module's MTTE measurement increased slightly but the general system execution performance bettered overall.

1. **Maintenance Cost Reduction:**

Through Planning Poker estimation technique the story points of reengineered classes decreased from 15 to 6 points.

The cost estimation dropped by 36.8% from 56.29 PM to 35.57 PM.

Agile cost estimation methods succeeded in determining the impact of the reengineering project.

**Conclusion:**

The study confirms that software reengineering delivers multiple benefits which include enhanced internal design quality and better performance alongside decreased maintenance expenses. The research demonstrates that software reengineering together with agile practices results in stronger control metrics and faster execution speed thus modernizing established software systems. Future research will apply these results to Analyze additional intricate systems as a means to increase the practical usefulness of the proposed methodological approach.