In their paper the authors compare the so-called ‘Cleanroom approach’ to a standard software development approach by conducting an empirical study. They let 15 three-person teams develop a software system consisting of approximately 1500 LOC. Five teams are using a standard development approach (group A) while the other ten teams use Cleanroom (group B). A possible Hawthorne effect is avoided by selecting members of A from a different course than those of B. The experience of the teams is balanced at an average of 1.6 years.

Cleanroom aims for both, an improved control of the development process and improvement in product quality. To accomplish those goals the entry of defects has to be denied as effective as possible. This is done by using a software lifecycle of executable increments in combination with formal methods for specification and design (i.e. structured specifications and state machine models) and non-execution based program development. Instead of testing their code by running it, programmers have to check it manually with the help of formal reading methods (i.e. stepwise abstraction, inspection, group walkthrough and formal verification). Statistically based independent testing is used to check whether a certain amount of reliability has been met.

In the first phase of the study the authors derive metrics to be taken by using the GQM approach. As goals they define to characterize the effect of Cleanroom on the delivered product, on the software development process and on the developers. Prior to the execution phase the members of group B get educated in the techniques and methods they are supposed to use. Each team can define its milestones. Every time a milestone is reached an independent testing team is evaluating the code and giving feedback. Additional data is collected by a questionnaire regarding individual experience a priori, a post development attitude survey, static source code analysis and operation system statistics.

Results show that group B met significantly more of the requirements than B. In addition their system is less sensitive to the operational profile which is useful since such profiles are changing over time most often. During the independent testing the code of group Bs teams was more correct but only on a significant level if duplicate failures are not counted. An explanation for this is that group B did not differentiate between more or less crucial components. The performance of the teams of group B had a significantly less wide range of performance than those of group A even though they consist of twice the amount of teams. This indicates that using Cleanroom makes the development process more predictable.

Due to less complexity and a lot of comments the code of group B is better readable than the one of group A.

Concerning the language used (Simpl-T) there is a strong correlation between the usage of if clauses and successful testing as well as a negative correlation between the usage of case clauses and successful testing in both groups. In group A a correlation between the use of procedure calls and successful testing and a negative correlation between the use of while-loops and successful testing was detected in addition.

Afterwards the members of group B were asked if they missed program execution. Most of them answered with yes even though most of them stated they felt comfortable with stepwise abstraction. The authors propose to let the programmers have a look at their code running during the independent tests. All teams of group B had a good feeling when asked for the effectiveness using offline-testing, spend less time online and made their deliveries in time. The programmers’ general experience correlated with the test results.

For further research the authors propose to test the Cleanroom approach on a bigger project.