Developer Effectiveness

I used to work in companies which were or are in a way of transformation to Agile, Devops, CICD etc. On the beginning they stored tasks, features in Word documents, sent bug reports on emails directly to developers and so on. I have seen developers coming to the office in the morning in a rush to investigate issue from production. Spending hours in stressful atmosphere (business calls for status every hour..) After fixing issue they had to prepare release, deploy it to the environment. Usually it was manual process to install package on the server. During the fixing they could destroy other working areas (no time for regression testing). Developers worked with huge systems which had a monolith architecture. They have spend too much time on fixing bugs. This time could be used for new features with the newest technology, reducing legacy code, implementing test automation and so on. Developers were frustrated, many of them left the company to find something more challenging. Quality of the product was poor, bugs from production occurs on a daily basis.

Later company hired consultants to contribute in transforming into agile. We got scrum masters, agile coaches, devops engineers, test automation engineers, new project managers. After few months Devops moved all code repositories to centralized git repository, build and deployments were moved to CICD tool that generates release on button click, which reduced a lot of time. They created common place with documentation, standards, how-to, etc (Confluence), that people knew where to find information. Test automation engineers created production like testing environments, build automatic regression tests including it to the deployment pipelines. Now after each bugfix automatic regression tests were executed. Developers got feedback after minutes from CI/CD tool instead of angry business. In general frustration decreased and quality increased. Developers got extra time which was saved due to the mentioned changes. New project managers comes with some nice ideas like every week sharing knowledge day. It was a day with pizza and some presentation about new technologies etc. Teams were sharing knowledge across company. They started moving monoliths to the microservices architecture with all industry standards like dockers, test automation on each phase. Code quality increased a lot. It is fully covered by unit tests. Most of the bugs are caught before deployment. People can focus on challenging tasks, coaching newcomers etc.