**Describe some of the best software development practices, and explain how these practices can positively or negatively impact the quality of the software being developed.**

There are a number of best practices to follow in software development. The following are just a few of them.

* Develop iteratively – By developing iteratively, critical risks can be resolved faster, thus saving on investment of time, money, and other resources. The initial iterations of a project enable early client/user feedback. Testing and integration are continuous. Developing iteratively provides a short-term focus for objective milestones.
* Manage requirements – Requirements are dynamic, so you can expect them to change throughout the software development process. One reason requirements might change is that a client/user's understanding of them can evolve over time. It is essential to gain agreement with the client/user on what the system should do, not how it is done. One should maintain forward and backward traceability of the requirements.
* Use component architecture – By using component architecture, one produces software that is reusable and widely available. Component architecture allows for improved maintainability and extensibility and promotes a clean division of work for development teams.
* Model software visually – By visually modeling the software, it improves software complexity management, captures component structure and behavior, hide or expose details as appropriate for tasks, and promote unambiguous communication.
* Verify quality – Quality is the characteristic of producing a product that meets or exceeds the agreed-upon requirements by some agreed-upon objective measures. Software problems are far more costly to locate and correct after deployment, so it is vital to develop test suites for each iteration for functionality, reliability, and performance.
* Control change – Explicit control of change helps prevent parallel development from degrading into chaos. Decomposing the architecture into subsystems and establishing secure workspaces allow development teams to be isolated from changes in other workspaces in order to focus on the issues at hand. Implementing an enforceable change control mechanism and version control system allows change requests to be prioritized, assess the impact of the change requests, and plan where to introduce the requested changes.