The agile development method and extreme programming are characterized by their less restrictive approach to software development. Agile development is a set of 12 principles that are defined by the Agile Manifesto, released in 2001. Firstly, the Agile development method can be described as a means of developing software that allows for easy refactoring and modification of existing systems. The Agile method follows the idea that working software is more valuable than comprehensive documentation. Following the agile method is consistent with the need for constant re-evaluation of a system's requirements as well as continued feedback from testers and clients. Agile development is largely user-focused, as the customer is directly involved with the product as it is being developed. Not only does this greatly influence the quality of the final product, but it also increases the customer's feelings of ownership over the software. Additionally, if the product needs to be brought to market quickly, it is easier to develop a functional, basic piece of software that can be improved upon in following iterations. However, Agile also presents some disadvantages. For example, if the customer does not have the time or resources to be directly involved in the project, then the user-focused nature of the project is lost. Additionally, Agile requires that the development team is completely dedicated to the project, as the project's development is not divided into easily manageable stages, rather divided into sprints where full features are delivered in a short amount of time. This is because requirements change so rapidly, and it is very easy to miss deadlines for new features. Agile projects are also much more efficient if the team is working near one another. The workflow of the project is likely to be impaired if the team's means of communication are unorganized or unreliable. The methodology defined by Agile is expanded upon in many other paradigms, such as Extreme Programming or Scrum.

Extreme Programming (XP) is an implementation of the Agile method, where functionality is added incrementally to the product, allowing for re-evaluation and refactoring. The value of XP comes from its ability to produce simple, manageable software that scales with the increasing size of the project, maintain higher morale within the development team, and deliver small features quickly and efficiently. The elevation of developer morale is caused by greater involvement from the programmers, increasing the amount of fulfillment experienced when delivering functionality, further motivating the team to deliver a quality product. Agile and XP both strongly adhere to the "YAGNI" (You aren't going to need it) principle, which states that excessive complexity is detrimental to the final product. Features are unit-tested before being added to the core application, allowing for flaws to be detected and de-bugged faster and more effectively. As with the principles of Agile development, refactoring is encouraged. Refactoring when following the Agile method comes at a much lower cost when compared to the Waterfall model, due to the fact that code is re-organized when the developers see the need for it, not when the product is nearing completion. Additionally, planning of the project is greatly decentralized, further increasing flexibility. It is important to note that XP still employs UML and other means of planning, although they are much less rigid and strictly adhered to. UML is simply a means of communicating the structure of a system, so its effectiveness is not lost with XP. The decentralization of the product's architecture has many benefits, such as keeping the engineers themselves in charge of the structure of the system. Often, dedicated software architects can be far removed from actual programming. This presents many issues. For example, a lack of recent programming experience can lead to misallocation of resources or possibly even choosing the wrong technology stack for the task. As newer and more powerful frameworks are released, it becomes more important for software architects to maintain knowledge of the current technological landscape. Therefore, Agile and XP often produce much simpler, more decoupled, and more scalable software.

Agile and XP, when compared to the Waterfall method (or the traditional method of software development), are best suited for specific circumstances. The choice of development strategy should reflect several characteristics of the project. The project's size, deadline, budget, requirements, and management should all be considered when making the decision of which methodology to adhere to. Projects that are larger, or require more flexibility, are typically developed under the Agile method. If the customer's needs are unpredictable and likely to change, then the flexibility of Agile provides the necessary freedom to modify the existing system without unnecessary costs to the project; along with the added ability to adhere more closely to the customer's needs. Moreover, smaller development teams are more suited for Agile than larger teams, because Agile requires much more effective communication and teamwork than the Waterfall model. Teams that are not centralized are not likely to be effective when developing separate features, while also adhering to deadlines. It is also important to consider the knowledge and skill level of the project management. Project managers are essential for guiding the team and maintaining the focus of the group, so an unorganized team leader could be much more detrimental when following the Agile methodology. Therefore, the Waterfall model is still a valuable strategy when developing an application, although its effectiveness is seen under different circumstances than Agile.

The Waterfall model is the largest alternative to Agile and is still widely employed in industry. It can be described as a much more plan-driven approach to building software. The architecture of the product is created by a team of software architects at the beginning of the project and the plan is strictly adhered to throughout development. Rather than being made up of smaller sub-projects that are added incrementally to the core application, the Waterfall model focuses on delivering the final product as one piece with full functionality. This comes at a cost though, as the capacity for the project to fail is much higher. This is because the project's value is not seen until the final product is released and all the functionality that was promised to the customer is implemented. The lack of communication with the customer does not allow for the team to reprioritize functionality or expand on it, as the requirements are typically included in the contract with the client. This lack of feedback can contribute to the increased rigidity of the product, and the likelihood that the customer's intent for the product is not realized. Thus, Waterfall is best suited for smaller projects that have rigid, non-volatile requirements as well as projects that do not require constant customer feedback. Waterfall's "all or nothing" approach also requires much less inter-team communication, as these interactions are limited to milestones in the project and are not required for concurrent development of separate features. Where Agile is considered an incremental approach to development, Waterfall progresses much more sequentially (divided into stages at the start of the project). Both methods are simply tools, and they both have advantages and disadvantages. It is the decision of the project leader to decide how the software will be built, and to consider the circumstances of the task.

An iterative approach to development can be described as the progressive refinement of functionality. The first iteration could potentially contain a problem (or many problems), so the following iterations take the base functionality and refine it to better fit the needs of the project. Agile and XP are both largely iterative, although the distinction to be made is that they are also incremental. Incremental development is the practice of splitting up features into modules, delivered at different stages in development. Agile implements both ideologies as features are added piece by piece as well as refined over multiple iterations. While Agile delivers pieces of the product that have room for improvement, it also adds full features to the core application without their value being defined exclusively by the final product. Iterative development is simply a piece of the Agile strategy, expanded upon with the incremental approach.

In conclusion, the Agile development method, the Waterfall model, and Extreme Programming, are all paradigms of software development that shouldn't be ignored. They each serve a specific purpose and the developers and management should do their best to make an informed, un-biased decision about which strategy to employ. Ultimately, the decision to employ one strategy over the other can greatly influence the productivity, morale, and success of the project and its developers.