**Spiral Model**

* The spiral model is a systems development lifecycle (SDLC) method used for risk management that combines the iterative development process model with elements of the Waterfall model. The spiral model is used by software engineers and is favored for large, expensive and complicated projects.
* When viewed as a diagram, the spiral model of software development looks like a coil with many loops. The [project manager](https://www.techtarget.com/searchcio/definition/project-management) designates the number of loops, which varies based on the project. Each loop of the spiral is a phase in the software development process model.

**Real-world examples of spiral model projects**

Various industries rely on the spiral model to iteratively improve projects. Examples include the following:

* Software development. Developers iteratively test software projects while adhering to feedback to guide improvements. This is especially true of mobile apps, where functionality changes rapidly and requires debugging to adhere to user and stakeholder expectations.
* Gaming. Game developers use this iterative model to test gameplay and improve graphics before a final product is released. Such refinements are based on customer feedback as well.
* Retail. E-commerce website developers use spiral modeling to continuously evolve and add new features to improve the customer experience based on consumer preferences and market trends.
* Healthcare. The spiral model is used to ensure electronic healthcare records systems meet industry standards and comply with existing regulations, such as the Health Insurance Portability and Accountability Act.
* Space. Space exploration systems, such as satellites and rovers, start as prototypes and undergo simulations for testing before being used in space. The spiral model guides their development to ensure they aren't prone to issues.

**Phases of the spiral model**

* When looking at a diagram of a spiral model, the radius of the spiral represents the cost of the project, and the angular degree represents the progress made in the current phase. Each phase begins with a goal for the design and ends when the developer or client reviews the progress.
* Every phase can be broken into four quadrants: identifying and understanding requirements, performing risk analysis, building the prototype and evaluating the software's performance.

**Identifying and understanding requirements**

Phases begin in the quadrant dedicated to the identification and understanding of requirements. The overall goal of the phase is determined, and all objectives are elaborated and analyzed. It's important to also identify alternative solutions in case the attempted version fails to perform.

**Risk analysis**

Risk analysis is performed on all possible solutions to find any faults or vulnerabilities -- such as running over budget or areas within the software open to different forms of cyberattacks. Each risk is resolved using the most efficient strategy.

Building the prototype

In the next quadrant, the prototype model is built and tested. This step includes architectural design, module design, physical product design and the final design. It takes the proposal that has been created in the first two quadrants and turns it into software that can be used.

Performance evaluation

In the fourth quadrant, the test results of the newest version are evaluated. This analysis lets programmers stop and understand what worked and didn't work before progressing with a new build. At the end of this quadrant, planning for the next phase begins and the cycle repeats. At the end of the spiral, the software is deployed in its respective market.