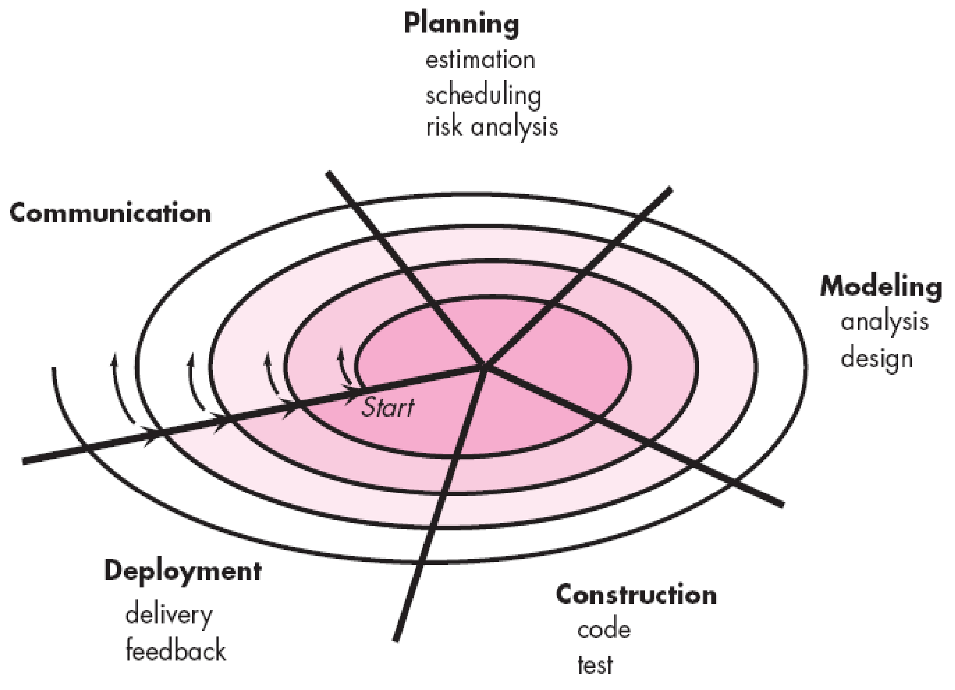
THE SPIRAL MODEL

The spiral model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model. It provides the potential for rapid development of increasingly more complete versions of the software.

Using the spiral model, software is developed in a series of evolutionary releases.

During early iterations, the release might be a model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

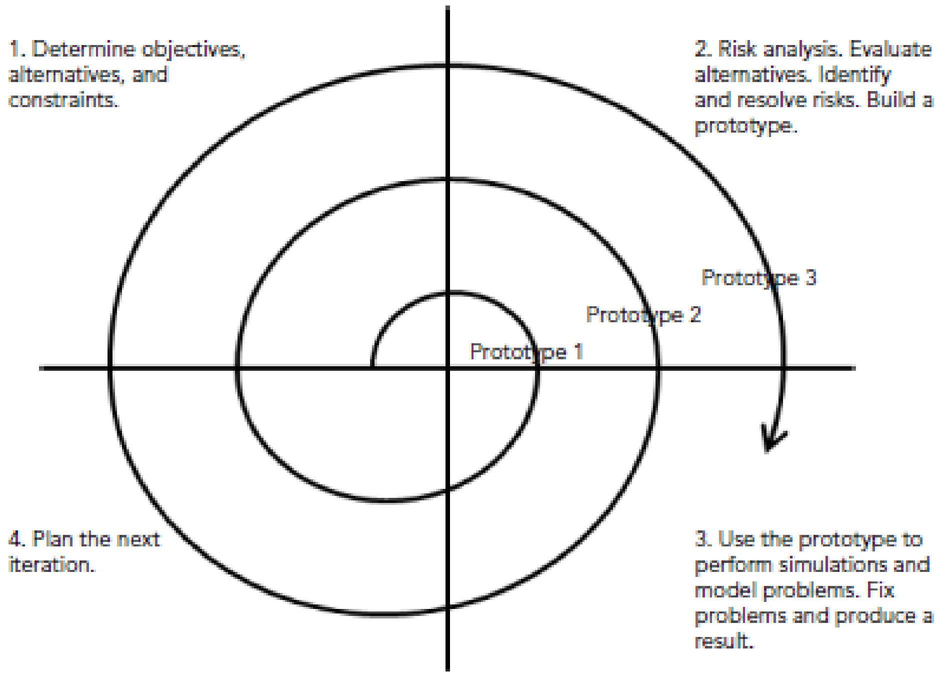


The first circuit around the spiral might result in the development of a product specification; subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software. Each pass through the planning region results in adjustments to the project plan.

Cost and schedule are adjusted based on feedback derived from the customer after delivery.

In addition, the project manager adjusts the planned number of iterations complete the software.

The spiral model uses a risk - driven approach to help project teams decide on what development approach to take for various parts of the project. For example, if you don’ t understand all the requirements, then you might use an iterative approach for developing them.


1. Determine objectives, alternatives, and constraints.
2. Risk analysis. Evaluate alternatives. Identify and resolve risks. Build a prototype.
Prototype 3
Prototype 2
Prototype 1
4. Plan the next iteration.
3. Use the prototype to perform simulations and model problems. Fix problems and produce a result.

The spiral process uses four phases.

In the first phase (which some call the planning phase), you determine the objectives of the current cycle. You define any alternatives and constraints on the objectives.

In the second phase (which some call the risk analysis phase), you perform a risk analysis to determine what the biggest risk factors are that could prevent you from achieving this cycle’s objectives. You resolve the risks and build a prototype to achieve your objectives. (Note that this may not be a program. For example, if the goal of the current cycle is to build requirements, then this will be a set of prototype requirements.)

In the third phase (which some call the engineering phase), you use the prototype you just built to evaluate your solution. You perform simulations and model specific problems to see if you’re on the right track. (For example, you might run through a bunch of operational scenarios to see if your prototype requirements can handle them.) You use what you learn to achieve the original objectives.

After this phase, you should have something concrete to show for your efforts.

In the fourth phase (which some call the evaluation phase), you evaluate your progress so far and make sure the project’s major stakeholders agree that the solution you came up with is correct and that the project should continue. If they decide you’ve made a mistake, you run another lap around the spiral to fi x whatever problems remain. (You identify the missed objectives, evaluate alternatives, identify and resolve risks, and produce another prototype.) After you’re sure you’re on the right track, you plan the next trip around the spiral.

Advantage

- Its spiral structure gives stakeholders a lot of points for review and making "go" or "no-go" decisions.
- It emphasizes risk analysis. If you identify and resolve risks correctly, it should lead to eventual success.
- It can accommodate change reasonably well. Simply make any necessary changes and then run through a cycle to identify and resolve any risks they create.
- Estimates such as time and effort required become more accurate over time as cycles are finished and risks are removed from the project.

Disadvantages:

- It's complicated.
- Because it's complicated, it often requires more resources than simpler approaches.
- Risk analysis can be difficult.
- The complication isn't always worth the effort, particularly for low - risk projects.
- Stakeholders must have the time and skills needed to review the project periodically to make sure each cycle is completed satisfactorily.
- Time and effort estimates become more accurate as cycles are finished, but initially those estimates may not be good.
- It doesn't work well with small projects. You could end up spending more time on risk analysis than you'd need to build the entire application with a simpler approach.

For those reasons, the spiral approach is most useful with large high - risk projects and projects with uncertain or changeable requirements.