Spiral model

- Barry W. Boehm is a mathematician and Professor of Software Engineering, Computer Science Department Director, USC Center for Software Engineering.

- He is known for his many contributions to software engineering including “the Constructive Cost Model (COCOMO), the Spiral Model of the software process, the Theory W (win-win) approach to software management and requirements determination and two advanced software engineering environments: the TRW Software Productivity System and Quantum Leap Environment.”

- The Spiral Model is a risk based approach which combines characteristics of evolutionary prototyping with the waterfall Model. As envisioned by Boehm, the Spiral Model is intended for large, complex projects with durations of 6 months to 2 years.
- This iterative model influenced Model-Based Architecture and Software Engineering (MBASE) and Extreme Programming.

- Today the Spiral Model has found a welcome home in the gaming industry. The large (>$3M) projects and continuously shifting goals of gaming systems development and resolutions achieved through evolutionary prototyping has created a demand for the process.
- For every spiral's iteration, there are four activity quadrants:

1 Determine objectives, alternatives and constraints:

System requirements are defined in as much detail as possible and include artifacts for functionality, performance, hardware/software interfaces, key success metrics, etc.

Alternatives such as build vs. buy, can we reuse existing components or do we sub-contract are examined.

Constraints such as cost, schedule and interfaces are addressed.

2.Identify and resolve risks, evaluate alternatives.

- All possible and available alternatives for developing a cost effective project are evaluated and strategies developed to determinate their use.

- Identify and resolve all the possible risks in the project such as lack of experience, new technology, tight schedules, poor process, etc.

- Resolve any found risks and uncertainties. Subset reviews may be commissioned to investigate other process models until all high risk situations are resolved.

3.Development and test:

Prototype the system from the preliminary design. Follow the usual pattern of create and review design, code, inspect code and test.

4. Plan the next iteration

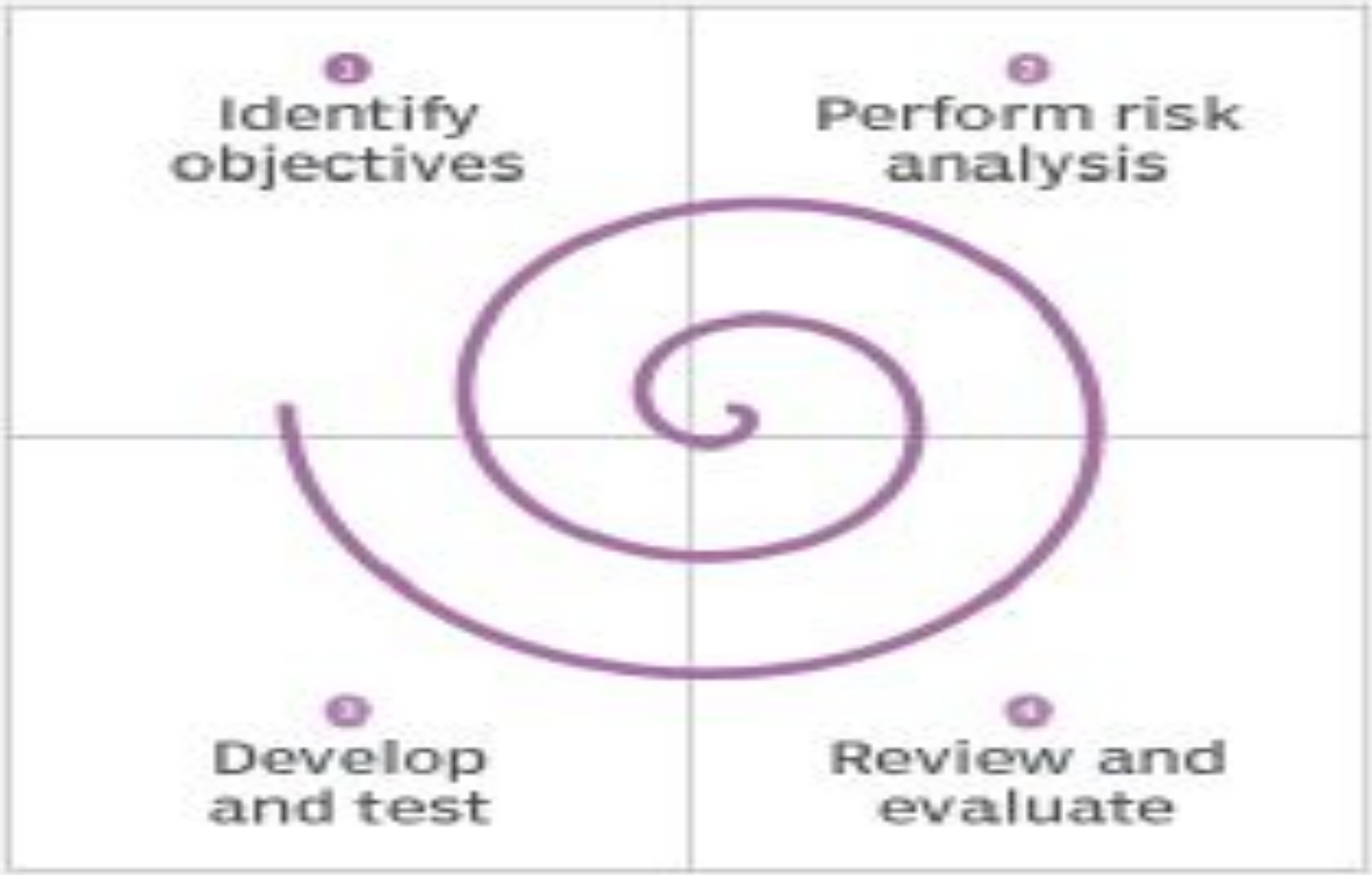
Review the first prototype for strengths, weaknesses, and risks.

Elicit the requirements for the second prototype.

Plan and design the second prototype;

- Create the project plan
- Document the configuration management plan
- Construct a test plan
- Devise an installation plan

Spiral model



Spiral model phases

- 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 evaluation of the software's performance.
- The overall goal of the phase should be determined and all objectives should be elaborated and analyzed.
- It is important to also identify alternative solutions in case the attempted version fails to perform.

- Next, risk analysis should be performed on all possible solutions in order to find any faults or vulnerabilities -- such as running over the budget or areas within the software that could be open to cyber attacks. Each risk should then be resolved using the most efficient strategy.
- In the next quadrant, the prototype is built and tested. This step includes: architectural design, design of modules, 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 utilized.
- Finally, in the fourth quadrant, the test results of the newest version are evaluated. This analysis allows programmers to 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 whole spiral, the software is finally deployed in its respective market.

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 looks like a coil with many loops. The number of loops varies based on each project and is often designated by the project manager.
- Each loop of the spiral is a phase in the software development process.

- The spiral model enables gradual releases and refinement of a product through each phase of the spiral as well as the ability to build prototypes at each phase.

- The most important feature of the model is its ability to manage unknown risks after the project has commenced;

- Creating a prototype makes this feasible.

Uses of the spiral model

- As mentioned before, the spiral model is best used in large, expensive and complicated projects. Other uses include:
- projects in which frequent releases are necessary;
- projects in which changes may be required at any time;
- long term projects that are not feasible due to altered economic priorities;
- medium to high risk projects;
- projects in which cost and risk analysis is important;
- projects that would benefit from the creation of a prototype; and
- projects with unclear or complex requirements.

Steps of the spiral model

While the phases are broken down into quadrants, each quadrant can be further broken down into the steps that occur within each one. The steps in the spiral model can be generalized as follows:

- The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure: (1) evaluating the first prototype in terms of its strengths, weaknesses, and risks; (2) defining the requirements of the second prototype; (3) planning and designing the second prototype; (4) constructing and testing the second prototype.

- The entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation and other factors that could result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and, if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.

Benefits of the spiral model

As mentioned before, the spiral model is a great option for large, complex projects.

The progressive nature of the model allows developers to break a big project into smaller pieces and tackle one feature at a time, ensuring nothing is missed.

Furthermore, since the prototype building is done progressively, the cost estimation of the whole project can sometimes be easier.

Other benefits of the spiral model include:

Flexibility - Changes made to the requirements after development has started can be easily adopted and incorporated.

Risk handling - The spiral model involves risk analysis and handling in every phase, improving security and the chances of avoiding attacks and breakages. The iterative development process also facilitates risk management.

Customer satisfaction - The spiral model facilitates customer feedback. If the software is being designed for a customer, then the customer will be able to see and evaluate their product in every phase. This allows them to voice dissatisfactions or make changes before the product is fully built, saving the development team time and money.

Limitations of the spiral model

High cost - The spiral model is expensive and, therefore, is not suitable for small projects.

Dependence on risk analysis - Since successful completion of the project depends on effective risk handling, then it is necessary for involved personnel to have expertise in risk assessment.

Complexity - The spiral model is more complex than other SDLC options. For it to operate efficiently, protocols must be followed closely. Furthermore, there is increased documentation since the model involves intermediate phases.

Hard to manage time - Going into the project, the number of required phases is often unknown, making time management almost impossible. Therefore, there is always a risk for falling behind schedule or going over budget.

When to use Spiral Methodology?

- When project is large
- When releases are required to be frequent
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- For medium to high-risk projects
- When requirements are unclear and complex
- When changes may require at any time
- When long term project commitment is not feasible due to changes in economic priorities

Advantages

- Additional functionality or changes can be done at a later stage
- Cost estimation becomes easy as the prototype building is done in small fragments
- Continuous or repeated development helps in risk management
- Development is fast and features are added in a systematic way
- There is always a space for customer feedback

Disadvantages

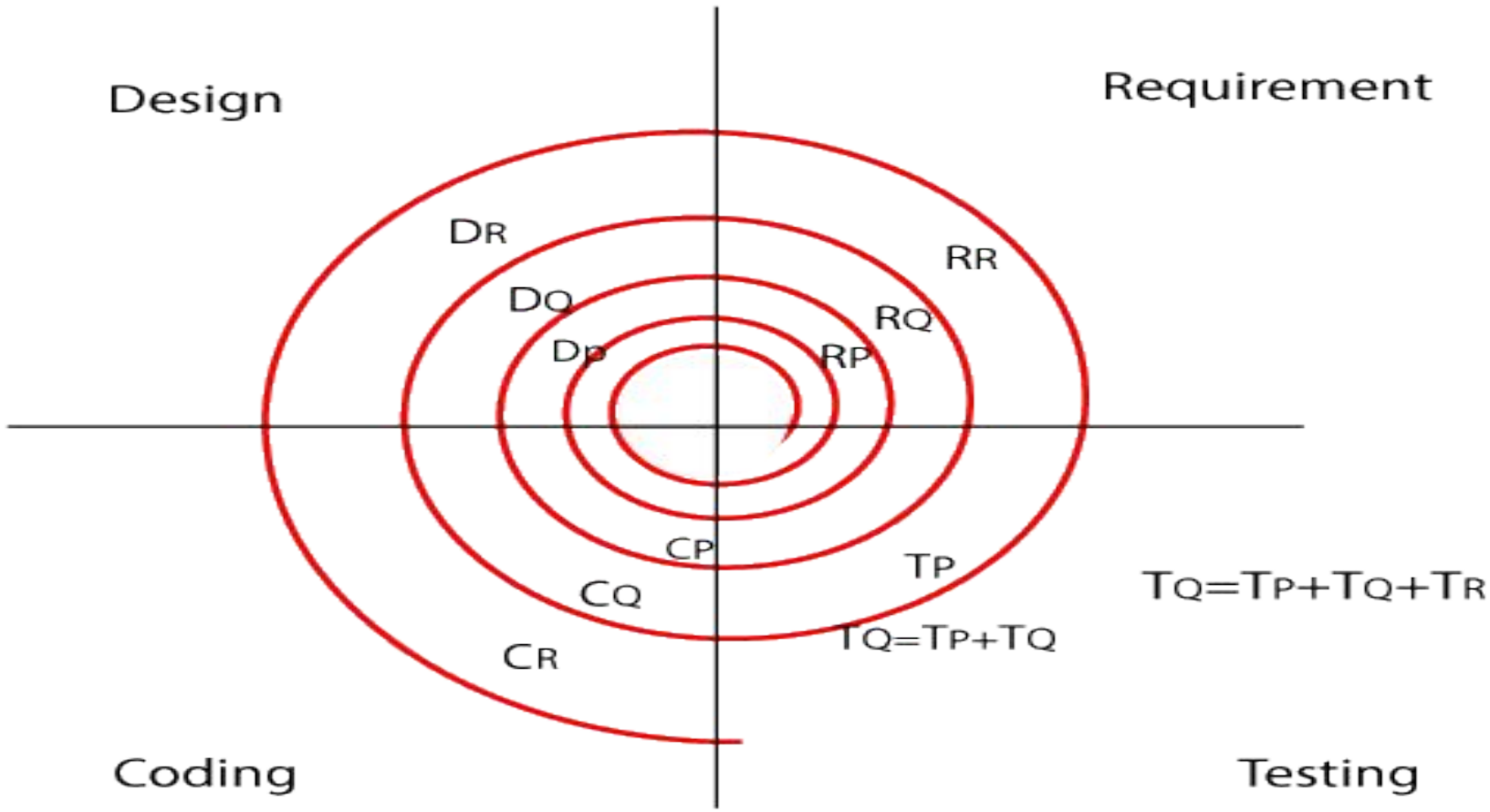
- Risk of not meeting the schedule or budget
- It works best for large projects only also demands risk assessment expertise
- For its smooth operation spiral model protocol needs to be followed strictly
- Documentation is more as it has intermediate phases.
- It is not advisable for smaller project, it might cost them a lot.

Example of the Spiral model

- Let us see one example for a better understanding of the spiral model:

- In the spiral model, the software is developed in the small modules. Suppose we have the application A and this A application is created with the help of different models as P, Q, R.

Examples of spiral model



Contd.

In the above image,

- **RP:** the requirement analysis of module P, similarly with RQ, RR.
- **DP:** Design of module P, and similarly with DQ, DR.
- **CP:** Coding of module P, and similarly CQ, CR.
- **TP:** Testing of module P, and similarly TQ, TR.

Contd.

- In the P module, we get the requirement first, and then only we design the module. And the coding part of module A is done when it is tested for bugs.
- The next module is Q, and it has been created when the module P has been built. We follow the same process as we did in module P, but when we start testing the module Q, and we check the following condition such as:
  - Test the Q module
  - The test integration of module Q with P
  - Test module P

Contd.

- After creating the module P, Q, we will proceed to the module R, where we will then follow the same process as module P and Q, and then test the following conditions:
  - First, check the module as R, Q, and P
  - Then, check the integration of module in the below order:
    R → Q, R and P → P and Q

Note:
Once the cycle continues for multiple modules, the module Q can be built only after the module P has been built correctly and similar for module R.

Contd.

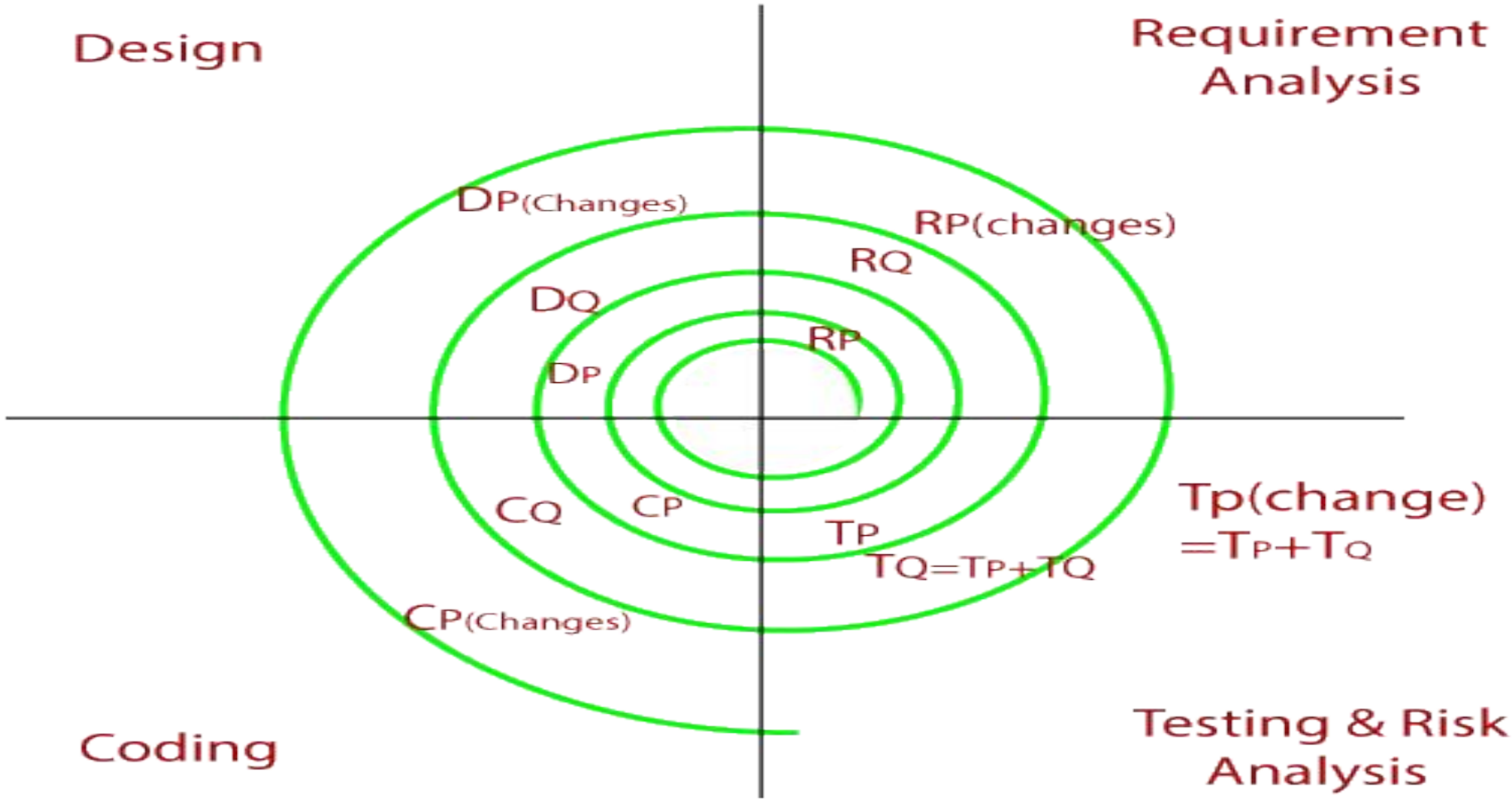
- The best-suited example for the spiral model is **MS-Excel** because MS-Excel sheet having several cells, which are the components of an excel sheet. Since we have to create the cells first (module P), then we can perform operation on the cells like split cells into half (module Q), merge cells into two, and then we can draw graphs on the excel-sheet (module R).
- In the spiral model, we can perform two types of changes, which are as follows:
- Major changes
- Minor changes

Major Changes

When the customer request for the major changes in the requirements for the particular module, then we change only that module and perform testing for both integration and unit.

And for this, we always prefer one new cycle because it may affect the existing modules. Major changes could be the functionality of the software.

Major Changes



Minor changes

Whenever the client request for the minor changes in the particular application, then the software team makes the smaller changes along with the new module has to be developed simultaneously in a single cycle.

And we never go for any new cycle or iteration because a minor variation does not affect the existing functionality, and it also takes the extra resource and time.

The minor changes could be UI (frontend changes).

Minor Changes