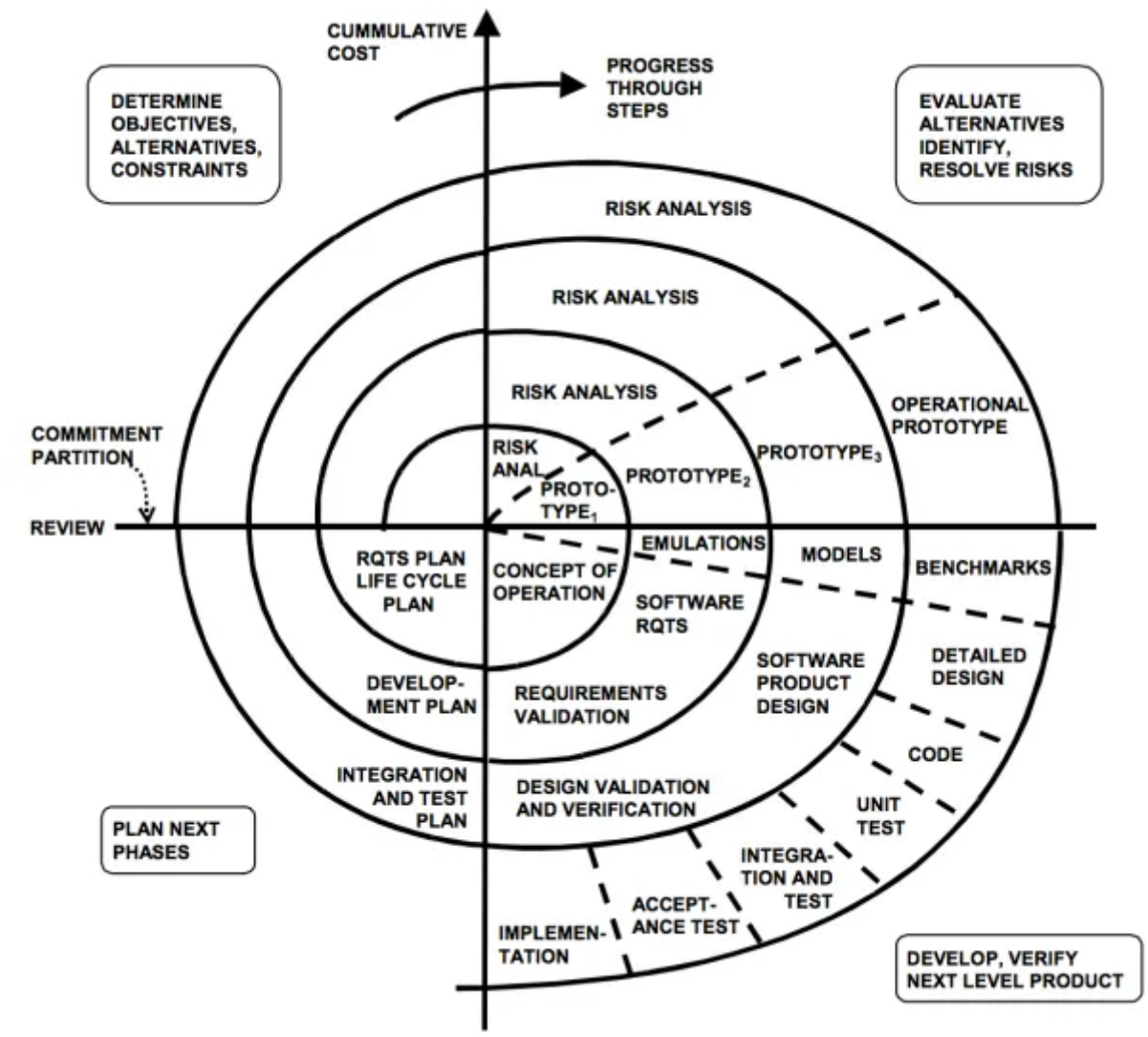
Spiral Model: Software Development For Critical Projects

September 29, 2016 | Andrew Powell-Morse | SDLC

Some more specific takes on SDLC include:

Rapid Application Development Test-Driven Development Waterfall Model
Iterative Model Extreme Programming Scaled Agile Framework
Agile Model Scrum Rational Unified Process
Big Bang Model V-Model Conceptual Model
Kaizen Model Kanban Model Software Development Life Cycle

The **Spiral Model** – first described by Barry Boehm in 1986 – is a software development methodology that aids in choosing the optimal process model for a given project. It combines aspects of the incremental build model, waterfall model and prototyping model, but is distinguished by a set of six invariant characteristics.



The original spiral model diagram. Credit – Barry Boehm, & Wilfred J. Hansen http://www.sei.cmu.edu/reports/00sr008.pdf

The Spiral Model is concerned primarily with risk awareness and management. The risk-driven approach of the spiral model ensures your team is highly flexible in its approach, but also highly aware of and prepared for the challenges they can expect to face down the road. The spiral model shines when stakes are highest and major setbacks are not an option.

1. Define Artifacts Concurrently (AKA "Plan everything, then re-plan those plans, then plan some more.")

In programming, an 'artifact' is any *thing* produced by people involved in the software development process. The spiral model suggests that all artifacts in a given project lifecycle should be defined fully from the start.

By planning each and every artifact of a project, your team reduces the potential for technical debt and other mishaps. For example, what if your software is reliant on an unusual, expensive or nearly obsolete piece of hardware? Or worse, what if after months of work you realize what you've been striving to build cannot actually be achieved? These types of issues can be disastrous.

Defining the scope of your **entire** project ensures you make the most of your time, and avoid potentially catastrophic scenarios.

2. Four Essential Spiral Tasks

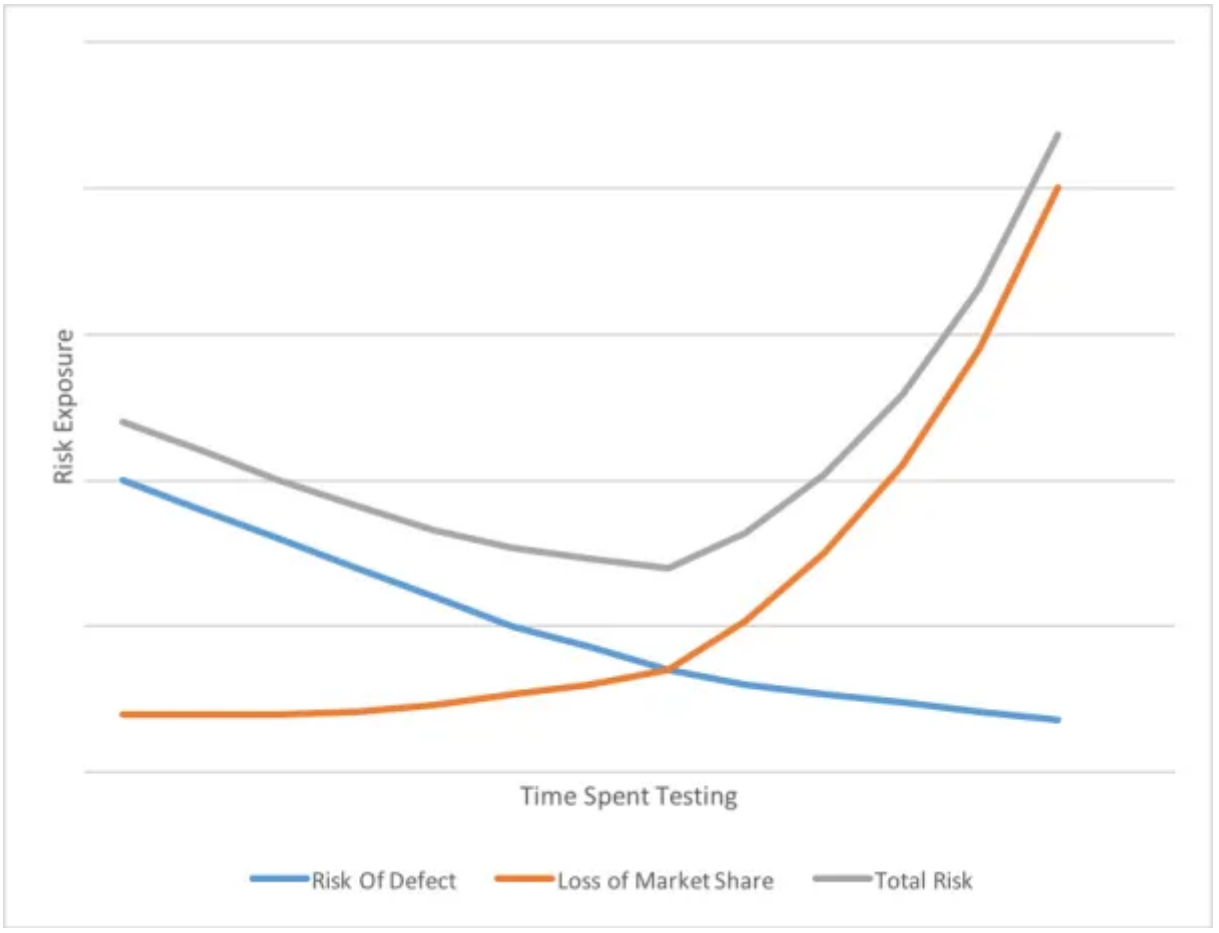
According to Boehm, every spiral model cycle consists of the following four tasks.

1. Consider critical-stakeholder objectives and constraints.
2. Elaborate and evaluate alternatives for achieving your objectives.
3. Identify and resolve risks attendant on choices of alternative solutions.
4. Stakeholders' review and agree to proceed based on satisfaction of their critical objectives and constraints.

The second characteristic of the Spiral Model – just like the first – is primarily concerned with mitigating risk and reducing the potential for setbacks. In this case, focusing on the wishes of critical stakeholders rises to the forefront. By performing these four activities, your team ensures it won't pursue options that don't satisfy the core needs of the project or present a high potential for risk.

3. Risk determines level of effort

Invariant three suggests that the effort allocated to a component of your project should be determined by the severity of risk associated with that component. Take the following chart for example:



We can see that after a certain point, the risk of losing market share far outweighs the gains made in testing. Yes, testing is important, and reduces the likelihood of releasing a buggy product, but speed in software development, speed is just as important (if not more important). Don't let your aversion to one risk hinder your ability to account for another.

4. Risk determines degree of details

Invariant four says the potential for risk should determine how much attention you pay to the details of the project you're working on.

Let's say you're introducing a new suite of features to your application. Seems like a no-brainer, right? More features = happier customers. Well…is that always the case? What if your existing customers become overwhelmed by a new interface, or miss the way your product behaved in the previous version? This situation could be pretty bad – you might have some people jumping ship. It would have been best to gather more data and implemented your changes more carefully.

On the other hand, let's say it's time to spruce up the language on your site to communicate your offering more clearly. Is this a critical component of your success? Yes. Is getting it wrong going to ruin you? Probably not. In this case, it's probably safe to throw a few different things up on your site, test them and see what sticks! Don't sweat the small stuff.

5. Use the anchor point milestones

The spiral model consists of three "Anchor Point Milestones." They are:

- Life Cycle Objectives (LCO)
- Life Cycle Architecture (LCA)
- Initial Operational Capability (IOC)

These milestones serve as intermediate checkpoints to keep the project moving full steam ahead in the right direction.

The LCO milestone checks to see if the technical approach to a project is well-defined enough to proceed, and that stakeholder conditions are met. If "Yes", continue. If no, abandon ship or commit to another lifecycle and try again.

The LCA milestone checks that an optimal approach has been defined and that all major risks are accounted for and planned for. If "Yes", continue. If no, abandon ship or commit to another lifecycle and try again.

The ICO milestone checks that adequate preparations have been made to satisfy stakeholders prior to launch. This includes the software, site, users, operators, and maintainers. If "Yes", its time for launch! If no, abandon ship or commit to another lifecycle and try again.

6. Focus on the system and its life cycle (Or "Software Isn't Everything")

Yes, the Spiral Model is a software development process, but that doesn't mean you can ignore non-software aspects of your business, or that every problem can be solved with a bit of code.

If you're experiencing troubles converting leads, that doesn't mean you need to build a sophisticated sales robot. Maybe it's just time to re-visit your sales tactics! More software is not always better, and you may find that throwing software at non-software problems only complicates them more. Focus on the underlying goals of the project and see to it that the solution you employ is the one best suited for meeting your win conditions.

So…what now?

While the Spiral Model can be a bit odd to define, one thing is for sure – it's a great way to keep your eyes on the prize and minimize the risk you assume when building something new. The invariants are great to keep in mind, but we've only just scratched the surface of this interesting software development life cycle tool. Check out Boehm & Hansen's official write-up below to learn even more about Spiral Model.

Spiral Development: Experience, Principles & Refinements – Boehm & Hansen

Share this:

Related

Agile Model: What Is It And How Do You Use It?
January 16, 2017
In "SDLC"

Waterfall Model: What Is It and When Should You Use It?
December 8, 2016
In "SDLC"

Iterative Model: What Is It And When Should You Use It?
December 15, 2016
In "SDLC"