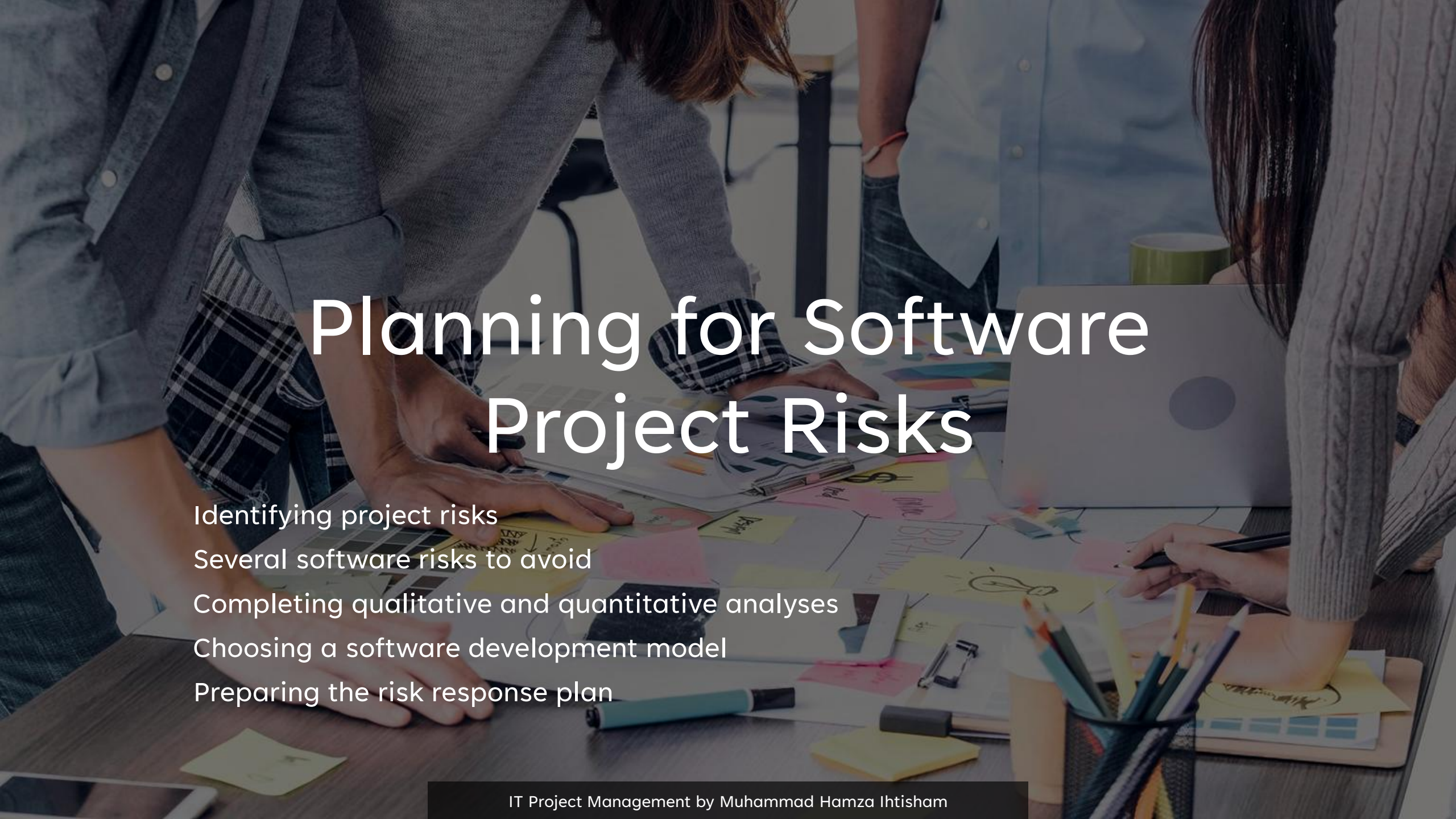
IT PROJECT MANAGEMENT

Muhammad Hamza Ihtisham



Planning for Software Project Risks

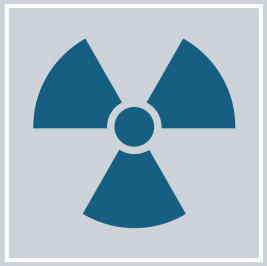
Identifying project risks

Several software risks to avoid

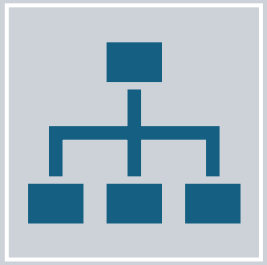
Completing qualitative and quantitative analyses

Choosing a software development model

Preparing the risk response plan



Pure risks: These risks have no upside, only a downside. Pure risks include things like loss of life or limb, fire, flood, and other bad stuff that nobody likes.



Business risks: These risks are the calculated risks you are concerned with in project management. A perfect example of a business risk is using a worker with less experience to save money on the project's budget. The risk is that the worker will screw up and your project will be doomed. The reward is that the worker will cost less than the more experienced worker and save the project some cash. An additional reward is that by challenging this employee you will encourage his or her growth and buy in on the project. This worker is more likely to be of greater value to you in future projects

Assessing business risks

Employees quit

Mistakes are made in the requirements gathering process

The software is full of bugs, errors, and failures

The scope of the project grows, but the budget (or the timeline) doesn't

The expectations of the project time, cost, and scope are not realistic to begin with

The project is larger than the capacity of the project team

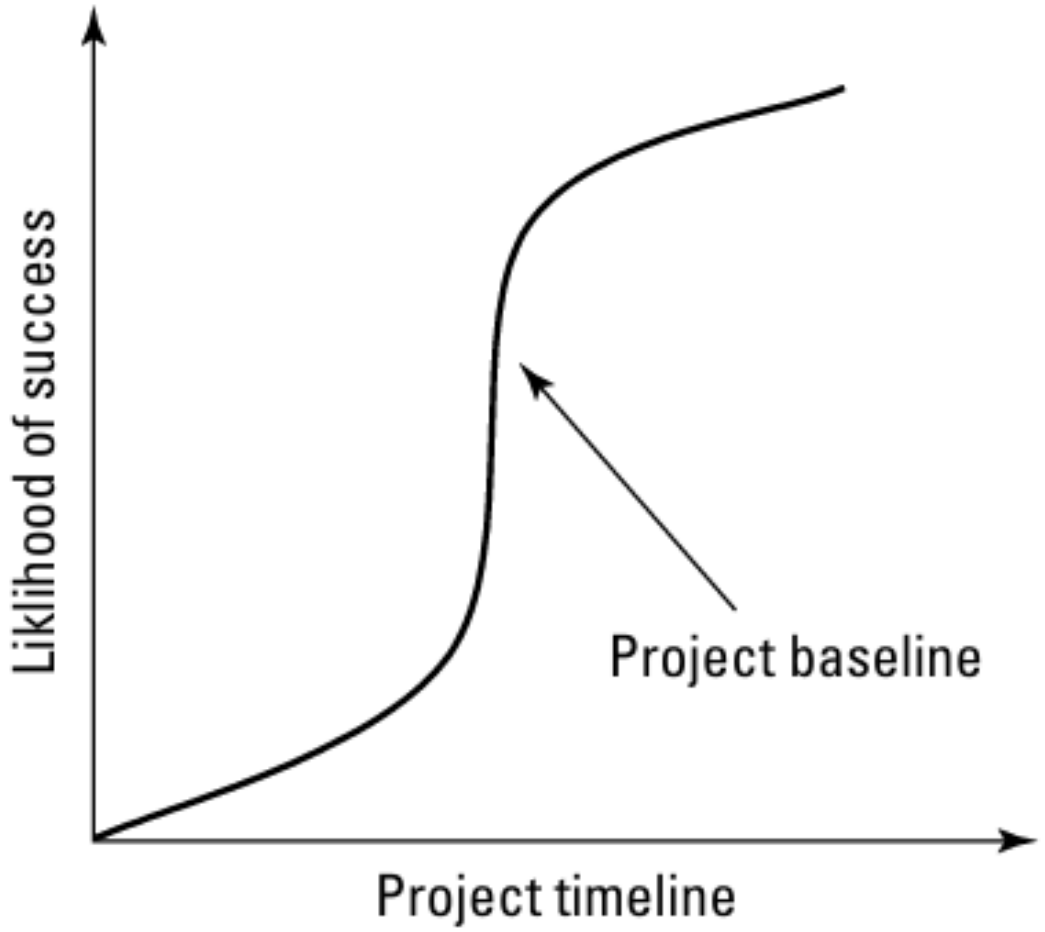
The project manager, sponsor, or other stakeholders are not as knowledgeable as you would hope

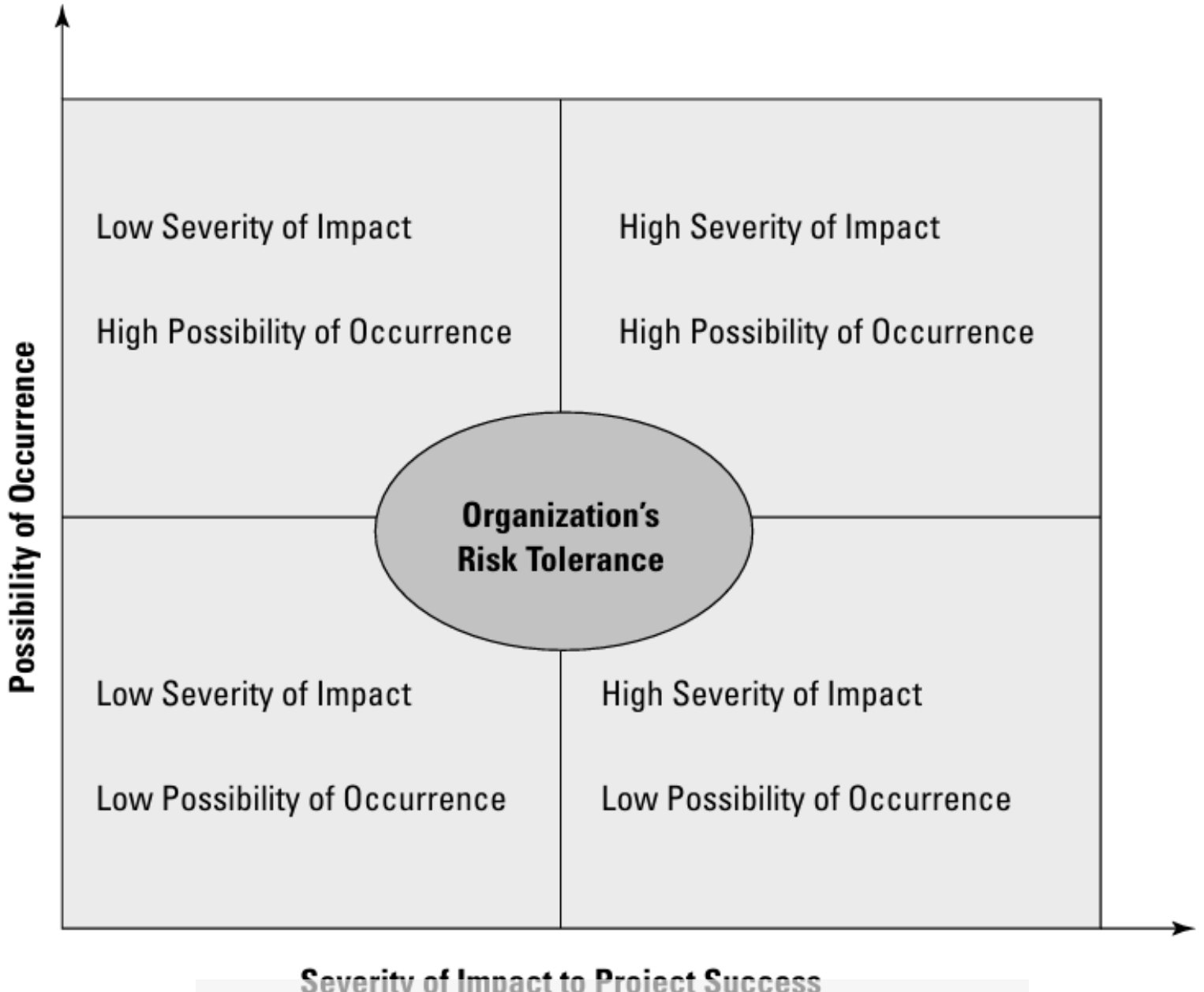
Accepting everyday technology risks with your software project

- Speed of technology surpasses demand for your creation.
- Delays in your schedule shorten the window for the demand of your software.
- Your programmers' ability to learn new programming languages and adapt to new development environments may threaten the project schedule.
- Your stakeholders may have a tough time explaining what they want the project deliverable to be.
- Because programmers are in demand, a programmer could leave your project team, putting your project at risk from loss of talent, time away from progress, and time devoted to getting a new developer up to speed.
- If your project has never been attempted before, you risk suffering from the first-time, first-use penalty.

Determining Stakeholder Risk Tolerance

- A person's willingness to accept risks is called his or her **utility function.**
- Figure shows an S-curve. As you can see, the higher the project priority is the lower the utility function is. In other words, the higher the project priority, the more likely you are to reduce risks. The number of acceptable risks will diminish.




Low Severity of Impact
High Possibility of Occurrence
High Severity of Impact
High Possibility of Occurrence
Organization's
Risk Tolerance
Low Severity of Impact
Low Possibility of Occurrence
High Severity of Impact
Low Possibility of Occurrence
Possibility of Occurrence
Severity of Impact to Project Success

Managing Risks in Your Organization

Identifying risks

- Regardless of your company's official risk management strategy, you can rely on qualitative risk analysis to get things moving. Qualitative analysis is the process of creating a risk ranking based on all the identified risks within the project.
- The best way to conduct qualitative risk analysis is for you to invite the project team and all the other key stakeholders to get together for a risk identification party.
- The method you use to gather information and identify risks is not as important as the fact that you are obtaining as much information as possible. For these risk identification exercises, put the emphasis on quantity. More is better.

Brainstorming

- During the first stage of risk identification, you should brainstorm. The crucial element of a brainstorm is spontaneity. Anything goes; no risk is too far out there.
- Include any conceivable risk that could threaten the project's success: server crashes, software failures, lost backup, weather, travel delays, meteorites, and so on. Ideally, your brainstorming session should be done in a big meeting with all the key stakeholders present. Trying to handle such an assessment via e-mail can really be a pain.

Following the Delphi method

- Another method of identifying project risks is to use the Delphi method.
- This allows stakeholders to anonymously offer their input into identifying risks.
- They can send their suggestions via e-mail to one person who will then consolidate the information into one document — without naming the source of the information.

Ranking risks

- After you and the key stakeholders identify all the risks you can think of, you need to rank them. We suggest ranking project risks by using a risk impact matrix. You can use two different approaches to risk ranking:
  - Ordinal: This assessment simply ranks risks as high, medium, or low. Most folks use ordinal for the first round of risk analysis.
  - Cardinal: When you use this ranking system, you assign scores with hard numbers, like .67 or .99.

Table 5-1 Sample Qualitative Risk Impact Matrix

Risk	*Probability*	*Impact*	*Risk Score*
Server crashes	Low	Medium	Low
Lack of developers	High	Medium	High
Firmware changes	Low	Medium	Low
Requirement to install service packs	High	Medium	Medium
Meteorites strike company headquarters	Low	Low	Low

Creating a Contingency Reserve

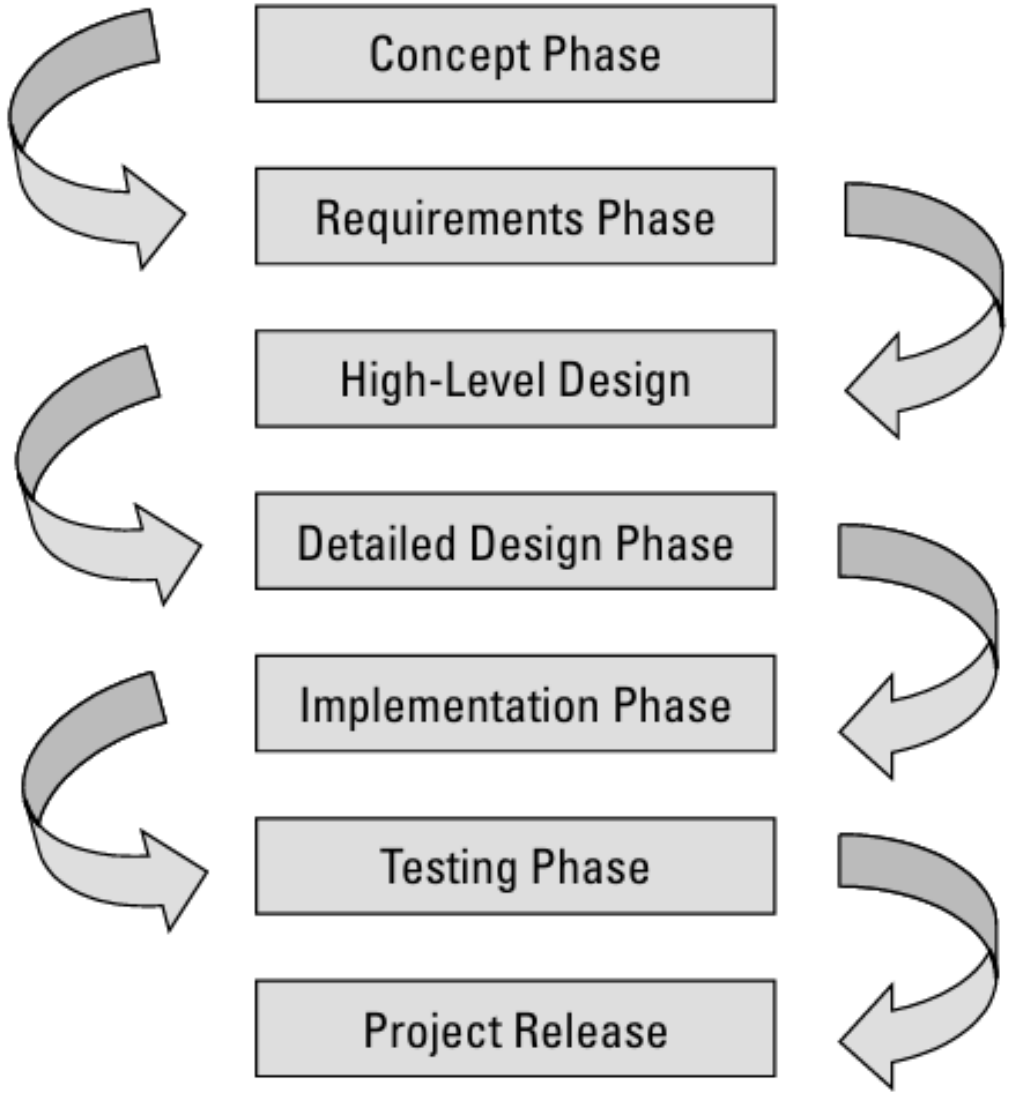
- Quantitative analysis also uses a risk impact matrix, like qualitative analysis, though a cardinal scale is mostly used here

Table 5-2 Sample Quantitative Risk Impact Matrix

Risk	*Probability*	*Impact*	*Risk Score*
Server crashes	.10	($5,000)	($500)
Lack of developers	.90	($80,000)	($72,000)
Firmware changes	.20	(10 days)	(2 days)
Requirement to install service packs	.70	($2,000)	($1,400)
Rebate from Manufacturer (risk opportunity)	.80	$1,000	$800
Contingency reserve needed			$73,100

Using Software Models for Risk Management

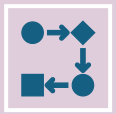
Using the waterfall model



Come up with a concept. In phase one, you and the stakeholders build the concept document, which explains the goals and constraints (such as time and cost) of the project, and a rough order of magnitude estimate (ROM) is created.

Determine the requirements. In phase two, using the concept document, the project team completes the document that details the requirements of the project deliverables. This document includes all of the systems, technologies, and technical interfaces for the deliverables. The requirements document is really the progressively elaborated version of the concept document.

Create a high-level design. You may hear the high-level design phase (phase three) referred to as the satellite point of view or the view from 20,000 feet. Whatever. The high-level phase consists of the big architectural building blocks of the software you're creating. The high-level design document defines how programmers will implement the requirements document.

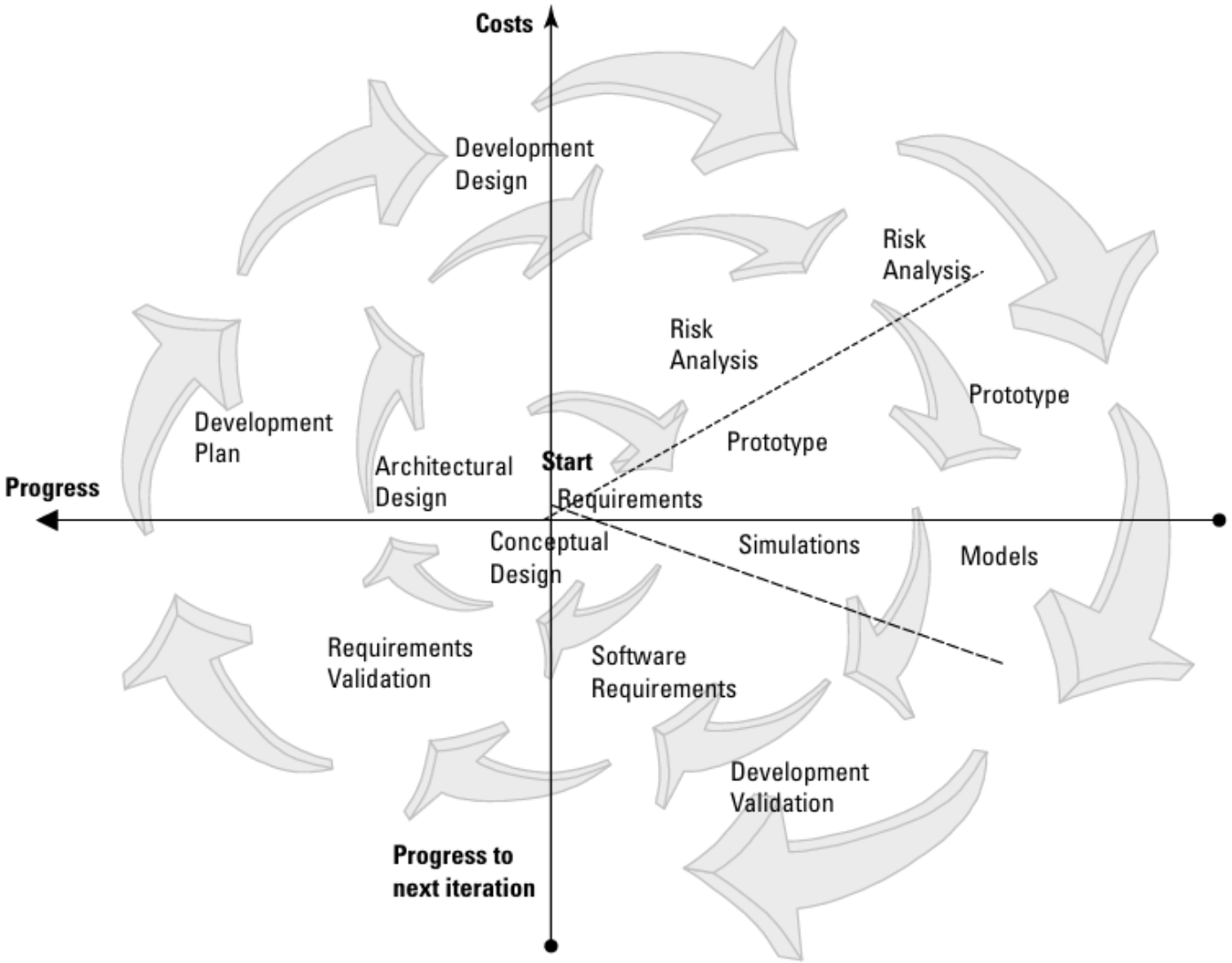


Narrow down the design to create a detailed design. Ah, now we're getting somewhere. In phase four, based on the high-level design document, you and the programmers can get into the details of how the application will be developed.

Code implementation phase. In phase five, it's time to follow the documents you've created with your project team by coding. This phase usually includes unit testing to ensure that what's being built follows all the documents created up until now.

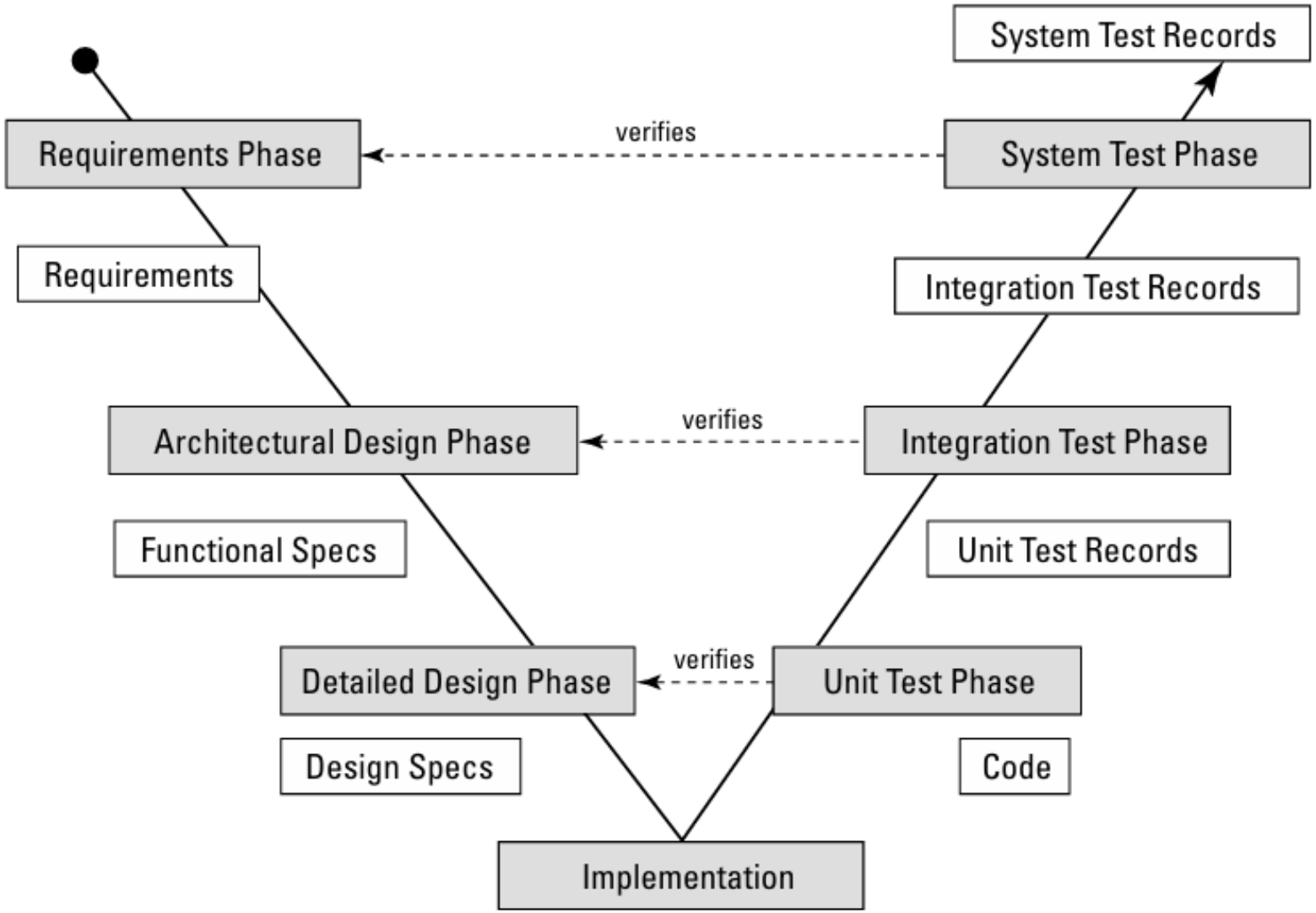
Testing phase. At this stage, you test the entire application to ensure that all the units are coded and working as expected — before your customers see it. This is the quality control phase. If you find problems, then you've got more work to do. The goal of this phase, of course, is to keep mistakes out of the customers' hands. When all is well, the deliverable is released.

Using the spiral model



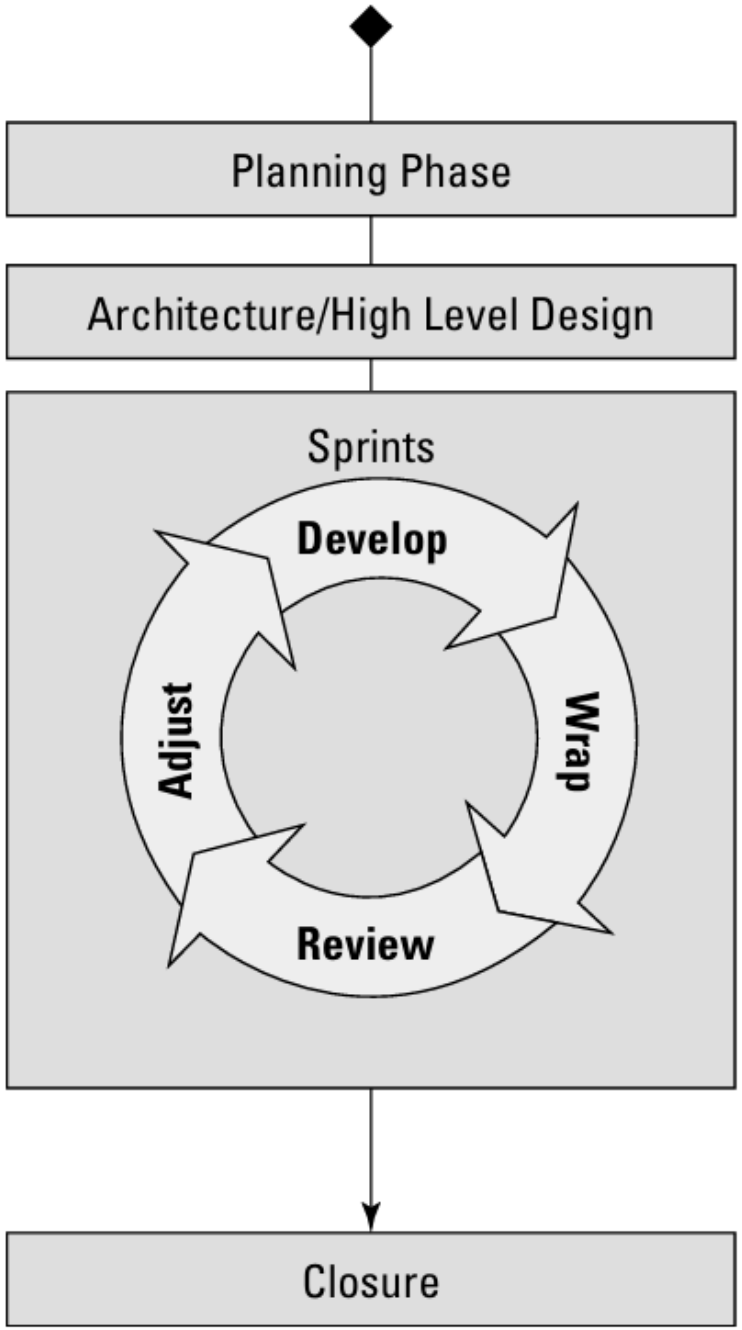
1. **Set a goal.** In the goal-setting phase, all the stakeholders work together to determine the goals, constraints, and alternatives for the project completion.
2. **Conduct alternative investigations**. The purpose here is to investigate all the possibilities to best achieve the project goals. In some organizations, you may create a feasibility study at this stage.
3. **Conduct risk management.** You and the stakeholders must examine, rank, and score all the project risks. Examining and ranking risks is a key activity for the spiral model because this process helps you and the project team to arrange the subprojects.
4. **Create the deliverables.** Based on the goals, alternative identification, and risk management, the project team creates a deliverable for this subproject. Each subproject delivers something that enables the project to move to the next iteration of the process. When you're starting out, the first deliverable may just be the verification of the goals, feasibility, and risk assessment.
5. **Plan the next iteration.** As the project moves forward, you plan the next iteration of the soft ware. This includes business like fixing bugs, adding new features, and making other improvements.
6. **Determine what to do next**. Based on the previous step, you determine what needs to happen next. After you make that determination, you move all the way back to step one and move through these steps with the next subproject.

Using the V model



1. **Set the project requirements**. The project requirements are identified and agreed upon by the project manager, the project sponsor, and other key stakeholders.
2. **Design the architecture.** The requirements are decomposed into functions and system components, and the project estimates for time and costs are updated. The refined estimates must be approved by the customer or the project's management team before the project moves forward.
3. **Elaborate on the architecture with a detailed design.** In the detailed design phase, the software's design phase is broken down further and designed, and a detailed design document is created. The detailed design document maps to the project requirements and specifies how the software will be created.
4. **Implement the code.** In the implementation phase, the code is implemented according to the detailed design document.
5. **Test individual units.** For the project to move forward, units are tested to confirm that the software works as described in the detailed design document. This is called the unit test phase. If the project passes the tests, the project moves forward. If not, the problems must be corrected and passed through the tests again.
6. **Test how everything works together.** In the integration test phase, you confirm that the software operates as the project stakeholders defined it (see Step 1).
7. **Test the whole system.** In the system test phase, compile the software and test the system. Successful testing of this phase allows the system to be released.

Using the scrum development model



1. **Set up a plan.** In the planning phase, the project team plans how to reach the project objectives. This phase includes prioritizing and making time and cost estimates and focuses on detailing the software's functions.
2. **Design the architecture.** In the architecture design phase, the team designs the software functions, breaks down the functions into units, and defines any additional features or components.
3. **Start sprinting.** Sprints are short, iterative bursts of software development. The result of a sprint is to reach a milestone within the project. Sprints typically last from one to four weeks. Multiple development teams may work simultaneously within the project, each working on their own sprints. Each sprint includes the following features:
    a. **Development.** The work packet is initiated. The development team completes the analysis, design, implementation, testing, and docu mentation.
    b. **Wrap.** The packet is wrapped. In other words, the work packet is closed. The code is verified as operational, and documentation of the work is created.
    c. **Review.** The project team reviews the work, points out and resolves issues, and adds items to the backlog for future resolutions. Risk is reviewed and mitigation efforts are introduced.
    d. **Adjustment.** The results of the review process are documented and, if necessary, compiled into work packets.
4. **Close out the project.** During the closure phase, the project results are tested and deemed accurate. The software is then prepped for release.