**Report on** [**Software Development Processes and Methodologies**](https://www.coursera.org/learn/software-processes/home/welcome)

**What is a requirement?**

How do you do requirements? Ask somebody learning how to program what the most difficult part of building software is, they'll tell you coding, or testing if they have a little bit more experience. But, ask a veteran software engineer and they'll tell you it's getting the problem right in the first place. This problem is compounded by a number of factors. First, building software is the most complex thing we've ever done. Building dams, rocket ships and nuclear reactors is hard. Don't get me wrong. But, in each of those cases, you're building something concrete. You can see it. You can touch it.

You can't do that with software. The fact that software is intangible, that it cannot be touched, makes anything much more difficult. Because of that property, it makes it very difficult to communicate exactly what we mean when we're talking with other people about our problem and it's a proposed solution because you can't see it. And on top of that, talking to the user.

They may not know what can be done to solve the problem. They might not even know what they want and two different users of the exact same system may want two different things. And, they can even call those two different things by one name, confusing everybody even more. So, what we do then is define a little bit of rigor into this very nebulous process.

We define our requirements specification. So, what is a requirement? Well, what is a requirements. Well, it's really two different things. First, it's a process. The process by which we create our shared understanding of both the problem that exists and eventually the needs of our supposed solution which we want to build that will solve that problem.

We create high level descriptions of all the concerns we hoped to address by our work with

a primary goal of developing a document which can clearly describe the details of what the system will do versus what it won't do. Additionally, it's very important that we capture the what and not the how. We hope to determine what the behavior of the solution will be without making any premature decisions that might affect our ability to design the solution.

The design solution will be decided but not at this stage. On the other hand, the requirement specification is also the product of that process. The requirement specification is the documentation that we produce from that process.

This can be an informal understanding, it could be a statement of work, it could even be a formal software requirement specification, an SRS document. IEEE is an international organization that has a computer society that has developed a template SRS which we're going to post.

You can take a look at what a formal document might look like. There are many reasons why this requirement specification process is important but the most widely cited reasons fall into two categories, engineering and economic.

The engineering argument is pretty simple. Spending time up front saves time later.

We make fewer mistakes, mistakes which often have far-ranging impact. When we spend a significant percentage of our work timeline on this planning stage, we do much better.

Don't plan anything, see your schedule and budget exploded.

Makes sense. But even with just 5% planning, you can see a significantly decreased overrun.

Now, notice, please, that it doesn't go to zero. Even spending 10-20% on requirements and scope only brings it down to 40% cost overrun with a long tail that looks to approach 35% but not much lower. Just another reminder that we aren't very good at estimating these kinds of things. Now, the other requirement for spending good time on requirements specifications and scoping is more microlevel.

We said before, most problems come from this stage and the longer they fester in the product,

the more costly they are to fix once we do find them. In fact, when the problems aren't found until the product has been deployed, you can see repair costs

as high as a thousand times the cost if the problem would have been found during the requirements analysis.

It's nearly an exponential increase in cost at every stage: design, to coding, to unit and integration testing, to system testing and again a thousand times at deployment. Handling requirements specification process well is crucial for the success of any software development project.

Given that most problems make it into your project during this phase, it's absolutely crucial that you spend the appropriate amount of time in this stage to ensure that the problem and solution are well-thought-out before beginning any kind of design or developing work.