Advice on Schedule and Meeting Deadlines

Working to deadline is a key element of this project and this class. The deadline is firm. Late projects will be accepted for only three days, and with significant penalties that you really don't want to have imposed. If you reach the deadline and don't have a product to turn in, it's a disaster.

How can you avoid this disaster? The main techniques are explicit risk control, design-to-schedule, and iterative design and implementation.

Explicit risk control

It sounds obvious and trivial, but it's important: Spend some time up front thinking specifically about what might go wrong, and how you can minimize your risk exposure. Risk is often related to uncertainty, and is often addressed by ordering tasks to gain useful information early. For example, if there are certain aspects of the system that you are less confident in than others, you should make sure those parts of the system are built or at least prototyped near the beginning, not near the end of the project.

Design to schedule

The project schedule should be an explicit, primary consideration in your design. If you're not confident of being able to incorporate a feature within the schedule, leave it out. If it is taking too long to implement some feature, find a way to do without it. A useful technique here is “timeboxing,” which essentially means that when some part of the project is taking longer than expected, rather than adjusting the schedule you find ways to scale back the design.

Iterative design and implementation

Iterative development combines risk control and design-to-schedule. Your motto should be, “build early, build often.” As early as possible, you should get into a mode where you *always* have a working version of the product, even if it doesn't do much. In fact, you should always have two versions of the product: The one you just built, and the last one that is known to work. Rather than assembling and testing the product near the end of the project, you should be continually adding to it. I suggest that by the end of week 2 you should be on a schedule of building your product at least twice a week, and by the end of week 3 you should be building at least daily, perhaps twice a day.

Iterative development particularly addresses the risk of missing the deadline: As soon as you start building working products, you have *something* to turn in, even if it's not much. From then on it's just getting better, and the question is not whether you will have something to turn in but how good it will be.

Iterative development can also be used to address other kinds of risk. If part of the project is considered high-risk but essential, it should become part of the product very early so that you can either gain confidence that it is ok or you can start as soon as possible to find an alternative. If something is considered high-risk and non-essential, you may want to put it off until all of the essential parts of the product are working.

Finally, iterative development is useful for exposing problems early. This includes integration problems (getting the pieces to work together), but also schedule problems since each person's progress becomes visible to other team members. If someone says “I'm 80% done but nothing is working yet,” don't believe it --- insist that the part that is done be integrated into the product.