What is Scrum?

A variation on Sashimi, an "all-at-once" approach to software engineering. Both Scrum and Sashimi are suited best to new product development rather than extended development. Sashimi originated with the Japanese and their experiences with the Waterfall model. They had the same problems with the Waterfall model as everybody else, so they adapted it to suit their own style. Realizing that speed and flexibility are as important as high quality and low cost they reduced the number of phases to four -- requirements, design, prototype, and acceptance -- without removing any activities, which resulted in overlap of the Waterfall phases. Then they made the four phases overlap. (Sashimi is a way of presenting sliced raw fish where each slice rests partially on the slice before it). Other companies took Sashimi one step further, reducing the phases to one and calling it Scrum. (A scrum is a team pack in Rugby, everybody in the pack acts together with everyone else to move the ball down the field).

Applying Scrum

For each Waterfall phase there are a pool of experienced people available, form a team by selecting one person from each pool. Call a team meeting and tell them that they have been selected to do an important project. Describe the project, include how long it's estimated to take, how much it is estimated to cost, how it is expected to perform, etc. Now tell them that their job is to do it in half the time, with half the cost, twice the performance, etc. Tell them how it's done is up to them and explain that your job is to support them with resources. Now leave.

Stand by, give advice if it's requested, and wait. Don't be surprised if a team member thinks the whole thing is insane and leaves. You'll get regular reports, but mostly you'll just wait. At somewhere around the expected time, the team will produce the system with the expected performance and cost.

How does Scrum work?

The first thing that happens is the initial leader will become primarily a reporter. The leadership role will bounce around within the team based on the task at hand. Soon QA developers will be learning how requirements are done and will be actively contributing, and requirements people will be seeing things from a QA point of view. As work is done in each of the phases, all the team learns and contributes, no work is done alone, the team is behind everything. From the initial meeting, the finished product is being developed. Someone can be writing code, working on functional specifications, and designing during the same day, i.e. "all-at-once". Don't be surprised if the team cleans the slate numerous times, many new ways will be picked up and many old ways discarded. The team will become autonomous, and will tend to transcend the initial goals, striving for excellence. The people on the team will become committed to accomplish the goal and some members may experience emotional pain when the project is completed.

Why does Scrum Work?

The basic premise is that if you are committed to the team and the project, and if your boss really trusts you, then you can spend time being productive instead of justifying your work. This reduces the need for meetings, reporting and authorization. There is control, but it is subtle and mostly indirect. It is exercised by selecting the right people, creating an open work environment, encouraging feedback, establishing an evaluation and reward program based on group performance, managing the tendency to go off in different directions early on, and tolerating mistakes. Every person on the team starts with an understanding of the problem, associates it with a range of solutions experienced and studied, then using skill, intelligence, and experience, will narrow the range to one or a few options.

Keep in mind that it can be difficult to give up the control that it takes to support the Scrum methodology. The approach is risky, there is no guarantee that the team will not run up against real limits, which could kill the project. The disappointment of the failure could adversely affect the team members because of the high levels of personal commitment involved. Each person on the team is required to understand *all* of the problem and *all* of the steps in developing a system to solve it, this may limit the size of the system developed using the methodology.