The mission:

Our mission is to create a system that enables developers to professionally deliver high quality and high value software in a repeatable fashion on a daily basis.

Important things to note within this mission. First is the word "enables" which is here because we want to create a supporting system, not an intrusive system. Second is the word "professionally". We want to encourage all developers to be professional in their work. Next "high quality" and "high value". All software created should possess these qualities. The software development system needs to encourage and enforce this. Finally note that this should become a daily and repeatable routine and as such needs a high level of automation.

The milestones:

In order to accomplish this mission, I propose a series of milestones that we can use to measure and proceed towards accomplishing this mission. At this time no dates are attached to these milestones. This is because while they are presented in as a series of events, they do not necessarily need to be accomplished in the presented order.

- Create a bi-monthly (or more frequent) program for education of developers.
    - Technology is constantly changing around us. Developers don't always have the time to research and learn the latest best practices or frameworks. By creating a standard scheduled education program, our developers can promote their own knowledge base.
    - The FBFS Learning center has many programs directed at business knowledge, this type of education could be coordinated with them for developers.

- Educate developers about proper logging.
    - It is important that all developers understand what it means to do logging within their systems and how to log meaningful messages in a way that is useful later.

- Implement logging practices into projects.
    - A common logging system will allow for the easier management of console statements. It will also create a more consistent environment for tracking down problems within the system if they do occur.

- Document and present idea for a corporate build system.
    - Software delivered to production needs to be able to be recreated. This needs to be taken out of the developers hands and done in a flexible automated fashion.

- Educate developer about ANT.
    - ANT (Another Nice Tool) is the de facto standard for creating build scripts for Java projects. It is vital that all developers understand how to use this tool as it is the foundation of the build system.

- Implement corporate build system.
    - Working together with other support teams, create the build system that all distributed projects will migrate to in the future.

- Implement first project to build system.
    - Find a project that has the band-with to migrate over to the corporate build system.

- Educate developers on dependency management
    - Projects rely on third party libraries to help solve problems. The developers should learn about how a dependency management system helps to reduce duplication across teams. This is also an important topic for asset management and the increased popularity of open source projects.

- Migrate a second project to the build system.
    - With the success of the first project migrated under our belt, we can take what we learned and apply it to a second project.

- Setup a local repository for use with a dependency management system (DMS).
    - By integrating the DMS into the build process a more consistent and governed approach will be created. A local repository will be needed to support the DMS and aid with auditing and asset management.

- Start a corporate discussion regarding some simple naming standards to be used.
    - Code standards, though a touchy subject, are important to define for all teams to use. They create a common lingua franca for all developers to communicate with within their code. This means that shifting developers across projects becomes somewhat easier.

- Educate developers on Unit Testing.
    - It is extremely important that testing happens during all parts of the Software Development Lifecycle, not just at the end. Unit testing is a technique which developers test small portions of their code in isolation from the rest of the code base.

- Implement automation support into the build system for unit tests.
    - The build system should run the developer written unit tests every time a build occurs. This helps to reduce the risk that what is being built will fail when tested by users.

- Implement automation support into the build system for code standard violation reporting.
    - Tools exist to enforce coding standards that are agreed upon by the company. The build can be set to not build if any are broken, or to just produce warnings. At first only warnings should be produced so developers have time to fix violations. Once a time period has passed these violations will stop a build from happening. This will help to reduce risks and increase quality.

- Introduce Continuous Integration Systems
    - Continuous Integration (CI) is the act of doing a build and receiving feedback as soon as new code is placed into the version control system. This milestone will educate developers about CI and start an project to evaluate the different CI systems available for use within Farm Bureau.

- Have discussions regarding static code analysis
    - Static code analysis tools exist that can automatically review code for any potential defects. This reduces risk and helps increase code quality. There are many different rules that can be implemented and a discussion needs to happen across the teams to decide which of these rules to use.

Monthly Reviews:

A review of the current systems in place should be happen monthly or perhaps quarterly. This will make sure that we are still providing value to the teams. It will also be used as a mechanism for feedback from the teams to help improve upon the system.

The image below summarizes the vision for how the build system as discussed above could work.