

# Agile Development

# The Context

- Heavy processes
- Unsuccessful projects
- Frustrated developers
- Frustrated project sponsors

# Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

# Reducing Risk

- Agile was designed to reduce risk by:
  - Shrinking the feedback loop
  - Making learning a part of the process
  - Focusing on what provides real value

# Individuals and Interactions

over processes and tools

# Working Software

over comprehensive documentation

# Customer Collaboration

over contract negotiation

# Responding to Change

over following a plan

# Agile Principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.**
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

# Agile Principles

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- **Simplicity--the art of maximizing the amount of work not done--is essential.**
- **The best architectures, requirements, and designs emerge from self-organizing teams.**
- **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.**