Traditional “Waterfall” style project management methods involve detailed up-front planning and design (often known as BDUF or Big Design Up Front). The features and scope are predetermined, and a schedule and associated costs are estimated. This works fine for construction projects, where details must by nature be predefined. For knowledge work, however, such as software development, research projects, most engineering projects, and other types of work that involve a level of uncertainty and high change rates, traditional methods fall short. In an environment where change is the norm and value can be improved based on information from new or unexpected sources, there is a more efficient and realistic way to manage. This is where Agile comes in. Although Agile-like methods were employed in Japan as early as 1986, it wasn’t until 2001 that a group of thought leaders in software development formalized the approach. This culminated in the Agile Manifesto, which cited 12 principles based on four overarching values that emphasized:

• 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 focuses on people working closely together in short iterations (generally two to four weeks), delivering some form of measurable value with each iteration and inspecting and adapting as appropriate. And throughout all this, the focus is on the product and the customer. Another difference about Agile is that instead of having a fixed scope and then estimating the schedule and cost, it involves working in fixed time and cost iterations, and estimating what features can be delivered in those iterations. In other words, it turns traditional project management on its head

**Collectively, this collaborative and iterative approach results in**:

• Faster value delivered to the user . . . because of the iterative process

• Reduced risk . . . because of iterative product delivery

• Reduced uncertainty . . . because of the product and customer focus

• Better decision making . . . because of the collaborative focus

• Increased trust . . . because of incremental value being delivered

• Fewer errors. . . because of greater communication

• Clearer accountability… because of delineated roles

With such benefits, it is not surprising that Agile has enjoyed such rapid growth in knowledge worker environments, extending beyond just software development where the idea originated.

Agile Roles With Agile projects, the role of the project manager must be reconsidered. This is because: Baselines are irrelevant (A baseline is meaningless if the whole concept of Agile is to plan and adapt the features to fit within fixed time and cost iterations.) Formal up-front requirements don’t apply (Customer needs must be understood, but detailed requirements are not all fixed up front; requirements evolve with learnings from each sprint.) The devil is in the details (Agile projects succeed or fail based on mutual understanding of customer needs and technical capabilities. Thus leaders of Agile projects need to understand the details.) It’s all about the product (Unlike traditional projects, where the focus is on managing the scope, schedule, and budget, Agile projects are all about the product as opposed to the project. Items like schedule and budget are fixed.)

• Fewer errors. . . because of greater communication

• Clearer accountability… because of delineated roles With such benefits, it is not surprising that Agile has enjoyed such rapid growth in knowledge worker environments, extending beyond just software development where the idea originated. 4 Agile projects are community-driven (In Agile projects, developers, analysis, product owners, and customers are in constant communication. If they’re not, then the Agile methodology is not being followed. Unlike traditional projects, the project manager is not the primary source of communication.) Agile projects are relatively low risk (With a large, complex project with lots of cross-team involvement and moving parts, Agile may not be the best approach. Agile is often used in software development and other knowledgework projects that can bear some sort of tolerance for uncertainty and a single team can work at a rapid pace.) With all this in mind, some may ask the inevitable question: What, then, is there for a project manager to do? It’s a valid question, and organizations have taken different approaches, from not having a project manager at all to using the project manager as a facilitator across the product owner, customers, and developers. In some organizations, the Scrum Master serves as the project manager.

Below are typical roles in an Agile project:

• Product Manager/Owner – determines the product vision and ensures that features listed in the backlog are prioritized and understood; provides User Stories (users/actions/benefits)

• Customer – monitors progress and provides input for valuable deliverables (Note: The Product Manager typically serves as the proxy for the customer when it comes to electronically recording comments and input)

• Development Manager/SCRUM Master/Project Manager – populates sprints from the project backlog and updates story points based on planning sprints; facilitates daily meetings and sprint demos

• Developers – are assigned to stories and make testing notes against stories; report effort for cost tracking purposes

• Resource Managers – optimize resource utilization across multiple projects and ensure resource availability on critical projects

• QA/QE Manager/Testers – focuses on quality assurance/engineering, including process improvements, testing, and measurement

Fear of Agile: Addressing the Concerns Since the beginning of the Agile movement, defenders of traditional methods have been skeptical. Some of their concerns are valid, while others are rooted in ignorance of true Agile methods. Ten common management fears are:

1. It won’t work for big, complex projects.

2. It’s too open-ended. We can’t predict costs. It’s “sanctioned scope creep.”

3. It sounds like “back of the napkin” design and planning.

4. It’s too “techie” focused.

5. Software developers don’t talk the same language as customers.

6. Customers don’t have time to get involved in planning.

7. We don’t want our customers to see our dirty laundry.

8. This “teamwork” approach doesn’t sound practical.

9. Daily meetings? Our employees will feel like they’re under a microscope.

10. It’s too rigid and inhibits individual creativity.

The truth is that Agile involves more planning than they realize – with each iteration in fact. And customer and business involvement is higher and communication is elevated, so the focus isn’t strictly on technology. But other concerns are quite legitimate. Overall, the following ten strategies can address typical concerns:

1. Make the strategy fit the situation (use the right methodology for the right job; Agile is best for a cohesive team working on knowledge work with a level of uncertainty.)

2. Reduce risk by focusing on business symptoms over solutions (don’t lose sight of the business problem being solved.)

3. Adopt a “see for yourself” policy to assess the user experience (before and after.)

4. Foster a systems-thinking approach (think beyond the software; think also of processes, upstream and downstream systems; stakeholder impact, and so on.)

5. Engage a business analyst to ensure that details aren’t overlooked (developers often lack an understanding of business details.)

6. Bridge the culture gap between technicians and customers (this can be done through training for developers on customer interaction or by engaging others to interpret and translate to bridge the gap between developers and customers.)

7. Embrace change, but manage it (assume features will change based on learnings, but have discussions about overall scope, and manage and log changes to the scope.)

8. Focus on product evolution, not project evolution; manage by prioritized features and release; have release targets (most sprints do not result in a deliverable to the customer, but provide some level of value; releases do result in customer deliverables; also, be sure to prioritize features and set milestones tied to releases.)

9. Gain management commitment to attend retrospectives and open demos (without management and customer involvement, the Agile methodology loses its power; gaining commitment is key.)

10. Focus on outcomes and value, not activities or hours (don’t make people feel like they’re under a microscope - focus on what is needed, not how to go about it; allow flexibility and individual creativity, and focus only on agreed-upon outcomes and value for each sprint - freedom with hours and creativity can balance out the heavy focus on achievement, and agreement on what’s achievable can reduce pressure.)

Planview Enterprise and Agile Planview

Enterprise has features to support organizations that use the Agile methodology. These include:

• Agile column sets when examining project schedules in the Work Manager. Built-in fields include:

• Sprint (the associated Sprint name)

• Story # - unique identifier code (can optionally link to a URL and/or other software)

• Story Status (e.g., Done, In Progress, Not Started, etc.)

• Sprint Start Date / Finish Date

• Story Points Planned (manually recorded)

• Story Points Planned Roll-Up (calculated automatically)

• Story Points Earned (manually recorded)

• Story Points Earned Roll-Up (calculated automatically)

• To-Do List and Notes functionality for story details

• Content Management for documenting and storing information from sprint retrospectives

• A Community page for discussions and team updates

• Resource authorizations for tracking time and cost

• Resource Reserves for high level resource demand planning

• Defined user roles for the various participants in the Agile methodology

• Custom reporting dashboards and Insight Analytics OLAP reporting tool, offering iteration velocity comparisons, weekly burndown charts, and end-ofiteration reports