
# Guide to Evaluate Projects for Use of an Agile Development Methodology



20 OCT 2011

# Contents

## 1. Introduction.

The Agile Manifesto, developed in 2001 is based on four values, "The four major values of agile methods are: (1) customer collaboration over contract negotiation, (2) individuals and interactions over processes and tools, (3) working software over comprehensive documentation, and (4) responding to change over following a plan."[1] These four values provide a paradigm shift from traditional software development processes. There is disagreement in the Agile community with some people stating that all projects can be run using an Agile methodology and others saying there are times when a more traditional approach is best. The matrices below show criteria from a number of sources (references 1-3, unless otherwise noted) that can be used to determine if a project is a candidate for Agile software development. Some of these criteria apply to the overall software development organization and some apply to individual projects. The Agile Methodology Decision Matrix at Figure 3 provides criteria for evaluating individual projects.

## 2. Organizational Characteristics.

Before reviewing projects to determine if they are candidates for Agile software development, the organization needs to be prepared to change from traditional business processes and procedures. Two primary changes that must be in place are having empowered development teams and lightweight documentation and process requirements.

| Organizational Characteristic | Agile | Traditional | Explanation |
|---|---|---|---|
| Empowered Development Team | Yes | No | Development team must be able to make design decisions without going through management gates. Requires management buy-in and customer buy-in to allow the development team to make key decisions. |
| Software Approval Process | Flexible | Rigid | Agile delivers capability in 2-4 or 2-6 week time periods (sprints) to provide features to the user. The entire SLDC must be Agile, not just the software development phase. Traditional SLDC processes and documentation must be tailored for Agile to be successful. |

**Figure 1 - Organizational Characteristics**

### 2.1 Empowered Development Team

Management needs to recognize that there will be a close relationship between the development team and the customer. Agile software development assumes that the development team has more autonomy than traditional software development teams.

> "The Agile methods embrace … self-organizing teams. This philosophy is a movement away from traditional command-and-control management and toward one that counts the team as an entity that has its own knowledge, perspective, motivation, and expertise. In this environment, the

[1] Rico, David F., Hasan H. Sayani, and Saya Sone. "Chapter 2 – Values of Agile Methods". *The Business Value of Agile Software Methods: Maximizing ROI with Just-in-Time Processes and Documentation*. J. Ross Publishing. © 2009. Books24x7.

> team is treated as a partner with management and the customer, capable of providing insight, affecting decisions, and negotiating commitments."[2]

Using traditional software development methods, some organizations have problems with users going directly to software developers to request changes to systems. With no framework or controls for this interaction it can lead to scope creep and undocumented changes. In Agile software development, interaction between developers and customers is encouraged, "The Agile methods all depend on regular and significant interaction between the development team and the customer."[3] Agile development will not lead to scope creep or undocumented changes if practices are in place which ensure the customer's discussions with the developers are related to the requirements set for the current iteration and that any requirements out-of-scope for the iteration are placed into the requirement backlog.

**2.2 Software Approval Process** Software approval processes, must be tailored for Agile development.

> Don't use agile if, "…deliverables must pass through a chain of approvals and tollgates: Approvals and tollgates are mini-bottlenecks that can slow the momentum of a project, momentum that agile methods are supposed to support."[4]

The focus of Agile development is interaction between the customer and the development team. Documentation is secondary to the personal interactions. That is not to say that documentation is not important, but it may come in a different order or format than traditional software development.

> "Agile methods are based on right-sized, just-enough, and just-in-time processes and documentation. Instead of inefficient documentation, they are based on frequent communications and they use a lightweight and highly adaptable project management framework called sprint or release planning. Agile methods produce user stories instead of requirements specifications, which take years to produce and will become instantly obsolete when the first line of code is produced. Also, all agile methods, processes and documentation are right sized to produce validated working software in 14-to 30-day sprints or iterations." [5]

---

[2] Koch, Alan S.. "Chapter 9 - Motivated Individuals and Self-Organizing Teams". Agile Software Development: Evaluating the Methods for Your Organization. Artech House. © 2005. Books24x7. <http://common.books24x7.com/book/id_14780/book.aspx> (accessed May 16, 2011)

[3] Koch, Alan S.. "Chapter 3 - Considering Your Customers". *Agile Software Development: Evaluating the Methods for Your Organization*. Artech House. © 2005. Books24x7. <*http://common.books24x7.com/book/id_14780/book.aspx*> (accessed May 9, 2011)

[4] Source: http://it.toolbox.com/wiki/index.php/Know_when_to_not_use_agile_methodology

[5] Source: Rico, David F., Hasan H. Sayani, and Saya Sone. "Chapter 23 - Agile vs. Traditional Methods". *The Business Value of Agile Software Methods: Maximizing ROI with Just-in-Time Processes and Documentation*. J. Ross Publishing. © 2009. Books24x7.

## 3. Project Characteristics.

Once the organization has made adjustments for Agile software development, individual projects can be evaluated. Asterisked (*) criteria were added by the author.

| Project Characteristic | Agile | Traditional | Explanation |
|---|---|---|---|
| *Data Sources Exist and are Available | Yes | Doesn't matter | Establishment of data sources, designing data bases or establishing interfaces, will not normally fit into a 2-6 week iteration. Once the data sources are established, then Agile methods can be used. |
| Availability of Customer | High – daily discussions on requirements and product | Low – weekly/monthly IPRs | Requires customer buy-in for the level of contact required, since daily contact is essential. |
| Team Stability | Highly stable | Moderately stable | At a minimum, core members must be stable. Focus is on tacit knowledge vice documentation; a stable team is essential since written documentation will be sparse. |
| Iterative Delivery of Functionality Acceptable | Yes | No | Iterative is not the same as the incremental delivery utilized by some organizations in the past. The organization must be prepared to develop, test and implement functionality in production at the end of each sprint (every 2-4 weeks)[6] |
| Requirement Stability | Low Stability (30-100% of requirements expected to change) | High Stability (0-29% of requirements expected to change) | Agile is best suited for changing or evolving requirement environments |
| Testing Effort | Low | High | If testing effort is high, such as requiring users in the U.S. and overseas to test every iteration, utilize traditional methods. |
| Skill/Experience/Training in Agile Methods | Yes | No | Team, including customer, must understand how Agile methodologies work. |
| *Number of Applications Involved | Low | Doesn't matter | Policy changes often involve numerous applications and multiple development teams which add complexity to managing the project. |
| Intermediate and Expert Agile Developers on the team | Yes<br>(50% or more of team is intermediate or expert in Agile) | No<br>(less than 50% of team is intermediate or expert in Agile) | This criteria shouldn't necessarily be used on the first few Agile projects an organization attempts, since expertise must be acquired through training and experience |
| Intermediate and Expert Software Developers on the Team | Yes<br>(50% or more of team is intermediate or expert in the chosen development language) | No<br>(less than 50% of team is intermediate or expert in the chosen development language) | Due to the fast pace of software development during sprints, the higher the percentage of experienced developers, the better. |
| Size of Team | Small to Medium | Doesn't matter | 10 or less on teams is ideal, at least as an organization begins utilizing Agile; larger teams can be managed using Agile, but experience is essential |

**Figure 2 - Project Criteria**

[6] Source: Wikipedia, retrieved 05/04/11 from http://en.wikipedia.org/wiki/Scrum_(development), "Characteristics" paragraph

**3.1 Data Sources Exist and are Available** – In the Decision Matrix, this criteria, along with the organizational criteria, are prerequisites that must be in place before considering a project for Agile development. Data standardization, data base design and establishment of System Interface Agreements must be done outside the Agile development cycle. While not impossible, it not likely to be practical to accomplish these activities in a two to six week iteration.

**3.2 Availability of Customer** – In the Decision Matrix, this is a critical criteria for Agile development. In the past, customers have not always been as tightly integrated into the development teams as required by the Agile methodology. The customer must be agreeable to the level of contact they will be required to have with the developers. "Agile management depends on an effective, day-to-day working partnership between the development team, the sponsor, and empowered users. If this partnership does not exist, or is likely to be weak, an agile project will face much higher risks than it should."[7] Some people go so far as to say, "If you cannot have access to the customer when you need them, agile fails. In agile projects there is no documentation to fall back to, the customer is the only source of information you have: without the customer an agile project will fail."[8] This author defined "access to the customer" as daily face to face meetings. Daily face to face meetings may not be practical for many customers. But, daily communications with the customers, face to face, via email and phone should be adequate to support an Agile development effort. The customer must agree to this level of interaction with the development team for a project to be a candidate for Agile development.

**3.2 Team Stability –** In the Decision Matrix, this is a critical criteria for Agile development. Agile development is based on people and their knowledge vice documentation, so the development team and the customer must not change during the development cycle. The team must be stable for a project to be a candidate for Agile development.

**3.3 Iterative Delivery of Functionality Acceptable** – In the Decision Matrix, this is a critical criteria for Agile development. Since software is delivered after each iteration, incremental build of the system must be acceptable to the customer. This requires the customer to be engaged in testing the software for each iteration. Iterative delivery of functionality must be acceptable for a project to be a candidate for Agile development.

**3.4 Requirement Stability –** In the Decision Matrix, this is a foundation criteria for Agile development. Agile was expressly developed to support uncertainty in requirements. Its strength is being responsive to and embracing changing requirements. This is not to say Agile cannot be used for projects with stable requirements, but it is best suited for situations with changing requirements.

**3.5 Testing Effort -** In the Decision Matrix, this is a foundation criteria for Agile development. Since testing is required during each iteration, which is two to six weeks long, the effort to test must be considered. If the testing effort is extensive, then traditional methods may be a better fit. An example would be if each iteration had to be tested by CONUS and OCONUS locations, the testing effort might be too cumbersome to coordinate within a two to six week time frame. Especially when you considered this will be done repeatedly, during each two to six week iteration.

**3.6 Skill/Experience/Training in Agile Methods** – In the Decision matrix, this is broken into two separate criteria, training is separated from experience. When an organization first moves into Agile development, they are likely to have few people with experience in Agile methods; training is considered a foundation criteria and experience is an influencing criteria.

---

[7] Source: http://it.toolbox.com/wiki/index.php/Know_when_to_not_use_agile_methodology

[8] Source: http://blogs.imeta.co.uk/TPeplow/archive/2008/11/28/when-should-i-use-agile-methods-on-my-software-project.aspx

**3.7 Number of Applications Involved** – In the Decision Matrix, Figure 3, this is a foundation criteria. Often implementing policy changes or other software requirements for an organization, involves more than one application. An example might be a policy change which requires software modifications to six legacy applications, plus new development work. This type of project would not be a good candidate for Agile development due to the difficulty of coordinating activities across the different development teams.

**3.8 Intermediate and Expert Agile Developers on the Team –** In the Decision Matrix, Figure 3, this is an influencing criteria. It is best to have more than 50% of the team either intermediate or expert in their skill level working Agile projects. As an organization begins to use Agile, this criteria may not be met during initial Agile project efforts, but as teams work on Agile projects and gain experience, this criteria will be met.

**3.9. Intermediate and Expert Software Developers on the Team -** In the Decision Matrix, Figure 3, this is an influencing criteria. It is best to have more than 50% of the team at either intermediate or expert skill level with the project's software development language.

**3.10 Size of Team** – In the Decision Matrix, Figure 3, this is an influencing criteria. Some authors state that the team should be no more than 10 people, but others say that team size does not matter. In the initial stages of using Agile, smaller teams will be better until experience is gained.

## 4. Agile Methodology Decision Matrix.

The Agile Methodology Decision Matrix, Figure 3, can be used to review an individual project to determine if it is a good candidate for using an Agile software development methodology.

**4.1 Agile Methodology Decision Matrix Categories.** The matrix shows the criteria from paragraphs 2 and 3 divided into four categories: Prerequisites, Critical Criteria, Foundation Criteria and Influencing Criteria:

- Prerequisites – All prerequisites must have an answer of "Yes" to proceed with the evaluation. The two organizational criteria (Development team empowered and Flexible SLDC with minimal documentation) must be in place before evaluating any project as a candidate for using Agile. The third criteria in this category, all data sources are available, applies to individual projects.

- Critical Criteria –These criteria are so critical to Agile software development that if there is a "No" answer for any criteria, go with a traditional methodology; the project is not a candidate for Agile methodologies.

- Foundation Criteria –A "No" in one of these criteria will add risk to an Agile project. The risk can be mitigated for a single "No", but if there is more than one "No" answer in this category, go with a traditional methodology.

- Influencing Criteria – These criteria apply to individual projects and have an influence on the success of Agile methodology for the project. "Yes" means the project is more likely to succeed, but "No" in any or all criteria can be mitigated by the project manager.

## 4.2 Evaluating a Project Using the Agile Methodology Decision Matrix

- All "Yes" answers in the Decision matrix mean the project is an ideal candidate for using an Agile software development methodology.
- Any "No" answer in Prerequisites or Critical Criteria mean the project is best managed using a traditional software development method and should not use an Agile methodology.
- One "No" answer in Foundation Criteria adds to the risk, but can be mitigated; more than one "No", use a traditional methodology.
- "No" answers in the Influencing Criteria can be mitigated by the project manager.

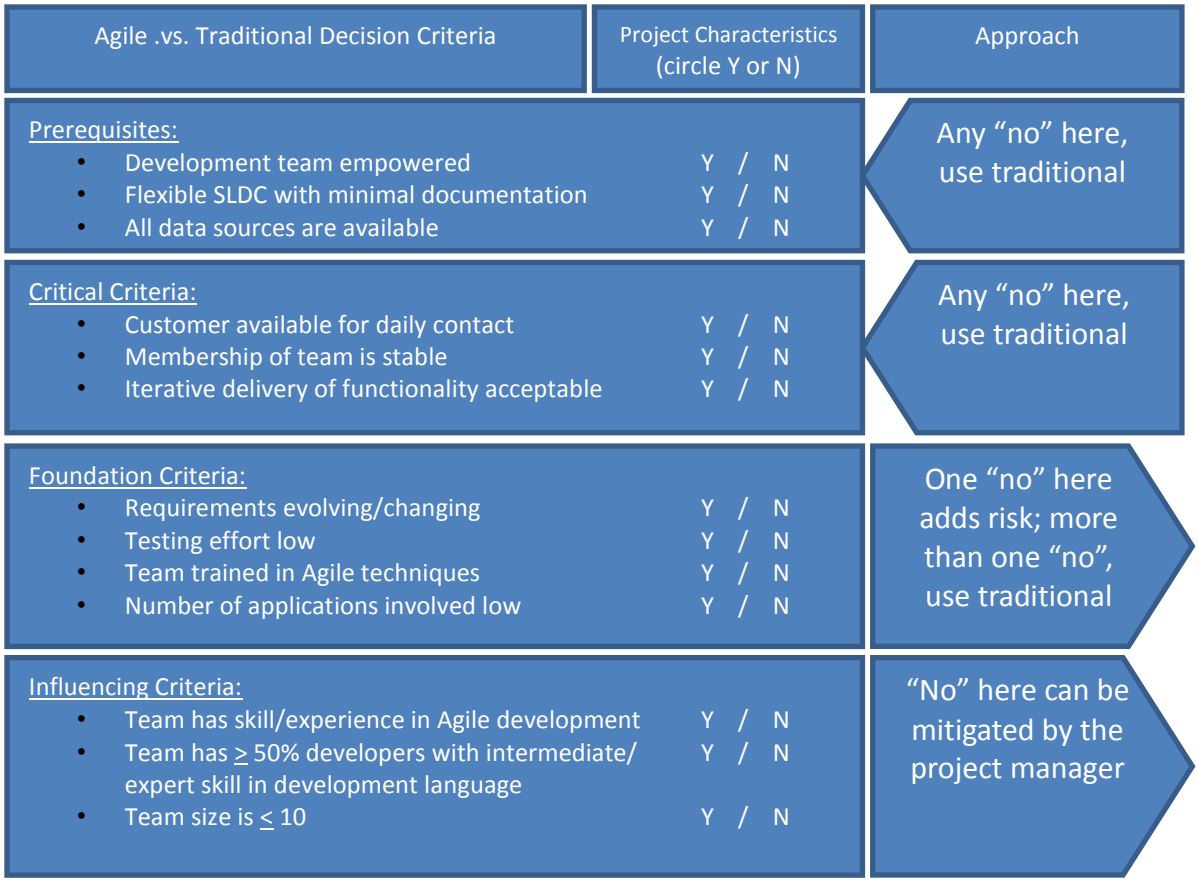


**Figure 3 - Agile Methodology Decision Matrix**

## 5. References.

1. "Agile Suitability Filters", Mike Griffiths, retrieved 04/27/11 from http://leadinganswers.typepad.com/leading_answers/files/agile_suitability_filters.pdf

2. "When should I use agile methods on my software project?", Tom Peplow's Blog, Retrieved 04/27/2011 from http://blogs.imeta.co.uk/TPeplow/archive/2008/11/28/when-should-i-use-agile-methods-on-my-software-project.aspx

3. "Know when to not use agile methodology", Toolbox.com Knowledge Sharing Communities. Retrieved 04/27/2011 from http://it.toolbox.com/wiki/index.php/Know_when_to_not_use_agile_methodology

4. Wikipedia, retrieved 05/04/11 from http://en.wikipedia.org/wiki/Scrum_(development), "Characteristics" paragraph

5. Rico, David F., Hasan H. Sayani, and Saya Sone. *The Business Value of Agile Software Methods: Maximizing ROI with Just-in-Time Processes and Documentation*. J. Ross Publishing. © 2009. Books24x7.

6. Koch, Alan S.. *Agile Software Development: Evaluating the Methods for Your Organization*. Artech House. © 2005. Books24x7.