

Experiences on Agile Implementations

Venkatesh Upadrasta
uvenkat76@yahoo.com

Table of Contents

Agile Methodology – A Software Engineering Approach.....	2
AN APPROACH	2
First principle: Resourcing	3
Second principle: Avoid Interrupts.....	3
Third principle: Customer availability & Engagement	3
Fourth principle: Delivery Model.....	3
Fifth Principle: Resource Management.....	4
Sixth Principle: Reduce Documentation & create user stories:	4
Seventh Principle: Iteration Planning Meetings	4
Eighth Principle: Architecting & designing.....	4
Ninth Principle: Standup Meeting	4
Tenth Principle: Extended Programming	5
Final Principle: Maintenance Phase	5

Agile Methodology – A Software Engineering Approach

Agile - A framework to function rapidly

It's called "Agile" for a reason. Be suspicious of anything that claims to be the "One True Way". It won't happen. You will have to adapt the process

- ❖ The Agile methodology for the Software delivery model uncovering better ways of developing software in short timeboxes not compromising quality
- ❖ Marries the best of both worlds
- ❖ Provides more importance to the business value of the application
- ❖ Give better visibility to the customer throughout the life cycle

'Agile' is the term given to a assortment of methodologies practiced widely for execution of software development projects that define the fundamental principles of frequent delivery of working software in iterations, customer collaboration and embracing change.

My personal anecdote on the Agile has surfaced successful during my past engagements (as a consultant) but one needs to be awfully careful while implementing them as a 'single point of failure' in the life cycle will end-up the project lead to disaster.

I will further, during the course of discussion, address some of the 'must to adapt' best practices to be formalized and introduced into projects for a customer centric and focused application following the agile methodologies.

AN APPROACH

DON'T EXPERIMENT AGILE METHODOLOGY IF:

1. You have to create long documentations on the requirements, design, codes and testing cases during each phase of the software life cycle
2. Deliverables needs to undergo a chain of approvals and tollgates
3. You use tools to create the software models but the tools shouldn't/doesn't translate them to the low end software code
4. You need to follow your traditional Software management processes and Procedures
5. Application users are NOT always accessible to you i.e., Your customers/users have limited involvement
6. **You want to work in a waterfall model** – In other words you are taking a serial approach for your project SDLC
7. The project team is not flexible on the working durations, overlaps & sifts
8. You don't have a strong team comprising of Business analyst's, technical Architect, Project Managers, Onsite/Offshore coordinators, Developers, etc. **Agile places a premium on having premium people for a project success**

As I move you further down the chain on Agile methodologies, i will be discussing in brief about managing projects –*MOUNTED ON AGILE PROGRAMMING*, mapping each to the software development life cycle phases translating - In contradiction, to one OR more of the above discussed points

I have seen cases of extreme failures of projects undertaken on AGILE, which have followed the famous, said 'Not followed ONLY 1 from the principles'



First principle: **Resourcing**

"It is better to have motivated, skilled people communicating well and using no OR some process than a well-defined process used by unmotivated individuals." [Cockburn 2002]

- Hire the smartest people you can find
- Plan and interview resources on the soft skills, interpersonal skills and their 'flexibility to work' not compromising on the technical/business skills. The methodology demands highly motivated individuals who has no constraints on the work timings, their flexibility to work on shifts, etc
- Give the team environment and support they need, and trust them to get the job done
- Ensure resources willingness to work on the alternate shifts. Provide them with the perks, recognitions they deserve for high **motivation**



Second principle: **Avoid Interrupts**

"Development is technical but not the least creative"

- Ensure that your team works best when they get into "flow" mode
- If they get broken out of flow, it takes ~15 minutes to get back in. The more interruptions, the harder it is to get back into the flow
- Try instituting Quiet Time for the programmers. Turn off the phone and e-mail, no meetings, etc
- If a question comes up, ping the requirement analyst- that's deemed OK, still it's on the project and on-topic which won't barge the flow like a random phone call will



Third principle: **Customer availability & Engagement**

- Ensure that your business users are available for the team **always**, for all the clarifications they require. Analysts can also be an effective go-between
- Ensure that the customer elaborates the user stories (Requirements in Agile terms are defined as 'User Stories') to an perceptible structure and extends the stories wherever possible
- Ensure that the customer works to define the next set of user stories after the other. The analyst teams should have the later set of user story ideas organized before completion of the former
- Ensure that the customer tests the application at predefined intervals after each iteration



Fourth principle: **Delivery Model**

"Welcome change in functionalities but not change in the business objectives"

- Do not enter into an engagement purely based on the profitability. Assess the complete project based on 'time to deliver', 'functional complexity', 'technical paradigms', etc and then enter into a contract. If you're pulling an engagement purely based on dollar value, you may have to give a little on service level. Print in mind that a cost-based contract might be appropriate for services like infrastructure management but not for specialized application development skills mounted on Agile programming. "You don't necessarily want to provide the cheapest heart surgery at the cost of quality"

- Deliver working software in an iterative model at frequent intervals in short cycles spanning from a couple of weeks to a couple of months based on the project size
- Most projects run 1-3 month cycles, with shorter iterations within that. Anything longer than that, you fall back into the dead zone
- *Actual users review actual software during the process:* Perform a User testing of the completed software and welcome changes
- Do not sugarcoat the news. "I'll be done in a week." Customer might not mind bad news as long as there's a good reason. But they hate feeling like they're being lied to

Fifth Principle: **Resource Management**

"With trust, freedom & supervision on staff makes the best software's"

- Alert, engaged staff is more effective than a tired, plodding staff
- Ensure that the team is spread across shifts (day & night shifts) not enforcing the shifts but on the resource willingness
- Team works not more than 10 hours a day with the next set of staff taking over from the former bridged by a coordinator
- Long hours are a symptom that something has gone wrong
- Avoid overtime. If unavoidable do not put in 2 weeks a row

Sixth Principle: **Reduce Documentation & create user stories:**

"Concentrate on working software rather than comprehensive documentation"

- Do not enforce creating comprehensive documentations during the project execution
- Create Short, a few sentences stories over the requirements
- Ensure that the Customers elaborate the stories to the Developers
- A story documented, must provide something of value to a customer. **Stories must be understandable**
- Developers DO NOT write user stories but reviews them
- Developers can ask for clarification at any point during the development
- Developers/Analyst(s) give high level estimate for the stories
- An estimate is always a guestimate in Agile, not an absolute commitment
- Group estimation of user stories is an important part of a project success

Seventh Principle: **Iteration Planning Meetings**

- Business Users should review the user stories understood by the analysts
- Customers reads out stories for the iteration
- Developers split the stories into tasks & discuss with the customer wherever required
- Any unsigned stories are deferred



Eighth Principle: **Architecting & designing**

"No ivory-tower architects/designers that hand something down from on high"

- "Architecture": Model that's hard to modify later. Have as little as possible
- Architect the application on a 'as less as possible' blocks and easy to use frameworks
- Create good designs initially, review and improve regularly & continuously
- Apply design cleanup at regular intervals rather than debt
- Alert the development teams on any of the architecture/design changes and engage them for suggestions

Ninth Principle: **Standup Meeting**

- Daily Meeting

- Less than 15-20 minutes
- Developers/Designers/Requirement analyst explain
 - What they've done from the last standup meeting
 - What they plan to do till the next standup
 - What got in the way of doing their work

Tenth Principle: **Extended Programming**

- **Pair Programming:** Two People creating the work products together, such that atleast one among the pairs will be concentrated on better evolving the solution
- Ensure that the coordinators; work on overlapping shifts with the teams, such that handover/takeover of the development happens in the Standup Meeting
- Ensure that the test teams are available as required for the development teams such that they can push their developments into testing immediately
- Compiling & Deploying code everyday/ everyweek are irksome to the developers. Reduce as much as the manual process and deploy all the sophisticated tools available, to reduce the boredom of the developers. Also deploy all the test management tools for the testing phase, since the iteration durations being small for Agile methodologies, one needs to repetitively run tests for a much inexpensive to fix bug former in cycle than later

Final Principle: **Maintenance Phase**

- Keep the system in production running while also producing new iterations
- May require new people in the team and changing the team structure.

Upadrista Venkatesh is currently heading delivery for couple of accounts in Wipro technologies. He has executed varied number of Agile based engagements in his past experience and has been the consultant for high profile customers with the goal to improvise their systems/processes with the Agile methodologies. During his experience he has successfully experiment his learning's and researches on Agile to his engagements which has gained him high popularity in the Industry.

He can be reached at uvenkat76@yahoo.com OR venkatesh.upadrista@wipro.com Work # 91 40 30793825; Mobile # 91-9849643233.