

Adopting Agile Methodologies of Software Development

By Nayan Jyoti Kar

Agile methodologies are challenging established paradigms of software development.

Agility in the business context is the ability of an enterprise to be proactive to changes happening in the environment in order to maximize the benefits. The key factors that determine the degree of agility of an enterprise are:

- Level of awareness – Keeping abreast of the changes and being prepared for the same.
- Innovation Quotient (I.Q.) – The percentage of those people whose ideas finally make it into practice [Skyrme, David 1998].

AGILITY: PROACTIVE VS REACTIVE

Being proactive has a definite edge over being reactive. Proactive means setting the pace for others to “catch-up” and reactive is exactly the opposite. Reactive enterprises often find themselves scampering to meet the challenges set by an innovative and aware enterprise.

The level of awareness and the I.Q. would determine whether an enterprise is proactive or reactive. The relationship between awareness, I.Q. and agility determine the state of an enterprise (Figure 1).

As the level of Awareness and I.Q. increases, the Agility also increases making the enterprise more proactive.

SIMPLE STEPS TO AGILITY

Agility can be attained using a simple two-step approach:

1. Handle external factors
2. Formulate Internal Policies

STEP 1: HANDLE EXTERNAL FACTORS

The external environment for an enterprise includes all the factors that directly or indirectly impact business operations (Figure 2). Some of these factors are:

Market: The market conditions are generally volatile and change dramatically. The reasons for change might be totally unrelated to the industry.

An Agile Enterprise is not caught unawares by the rapidly changing market scenario. It embraces the change and devises ways to use it to its own advantage. Sometimes, an Agile Enterprise introduces innovations that change the market scenario. This results in the

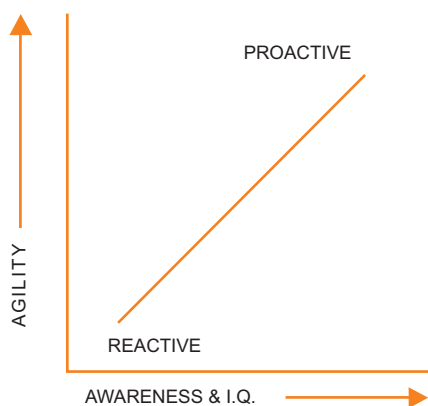


Figure 1: Relationship between Awareness, I.Q. and Agility
Source: Infosys Research

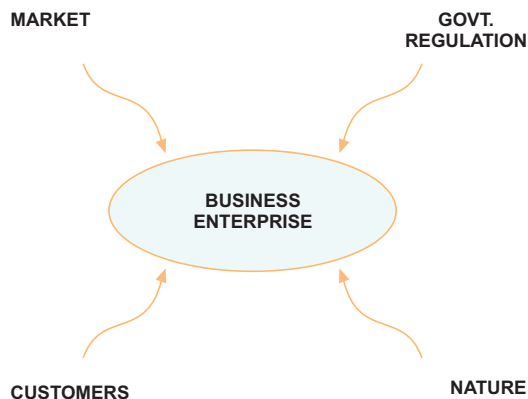


Figure 2: External Factors Influencing an Enterprise
Source: Infosys Research

realignment of the business plans of its competitors.

Customer: The customer’s role has gone through rapid changes in recent times from being a mere end-user to playing a more active role in the development process for the product leading to “co-creation of value”.

An Agile Enterprise welcomes the customer feedback and tries to incorporate them in the product. It also strives to foresee some of the changing business needs and provides suggestions for the same to the customer. This adds value to provider-customer relationship and opens new avenues for future business.

Government Regulations: New laws could be enacted or old ones modified over time that may impact the business. Usually, such changes are notified well in advance by the regulatory authorities. Awareness is the key with respect to this factor.

Natural Calamities: Natural calamities can be dealt with by having a business continuity plan and appropriate disaster recovery initiatives to

prevent a complete shutdown of operations and ensure that critical activities can keep on running in face of such a situation.

STEP 2: FORMULATE INTERNAL POLICIES

The Business Enterprise is a synergy of various smaller units performing different functions. Agility at the enterprise level can be achieved when all these units themselves become Agile. This can be achieved by formulating policies to govern the functions of the different units.

It is very important to formulate a flexible policy that evolves over time and a policy that acts as an enabler and not as a limiting factor.

There are various agile methodologies that provide a framework for formulation of such policies. These methodologies are primarily developed keeping software development as focus. They can be applied to other enterprises with minor modifications. It is important to note that there is no “best-fit” methodology for an enterprise.

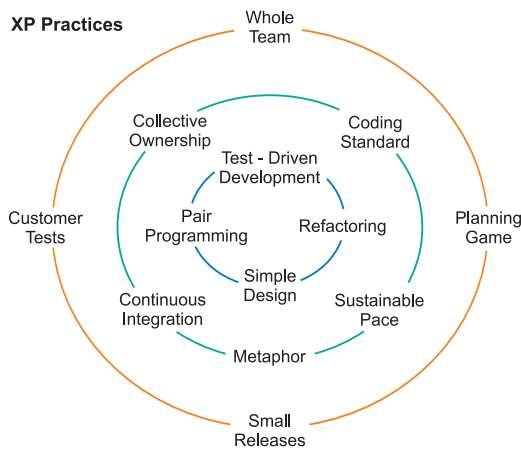


Figure 3: Core XP Practices
Source: <http://www.xprogramming.com>

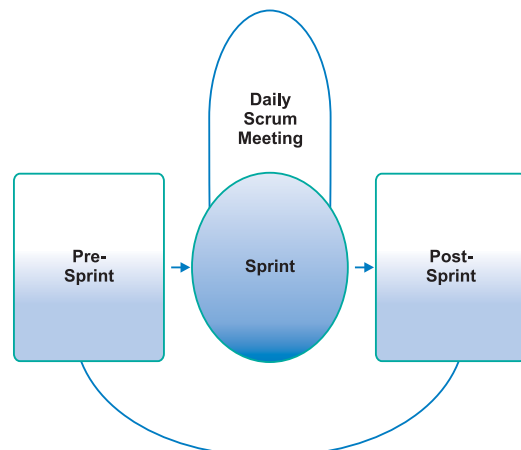


Figure 4: Three Stages in Scrum
Source: Conceptualized from <http://jeffsutherland.com>

POPULAR AGILE METHODOLOGIES

Extreme Programming (XP)

Extreme programming was developed by Kent Beck when he introduced it in his book “Extreme Programming Explained”. It has been one of the most popular agile methodologies owing to its simplicity.

“Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.” [Jeffries, Ron 2001]

The core XP practices which form the heart of this methodology are (Figure 3):

Whole Team: The team comprises of all the contributors - the development team (including a lead), the customer and the business analysts. The team together determines

- What needs to be done?
- How?
- What is the time frame?

This collective activity ensures that future efforts would be in the right direction.

Planning Game – Planning game lays the outline for the development of the product. It is a characteristic of almost all Agile methodologies. The activities involved are:

- Short iterations of 3 weeks
- Frequent Plan updates
- “Story” assignment (a story is a particular requirement displayed on a 3 X 5 card)

Small Releases – Frequent releases with partial functionality are delivered. This enables the customer to do an interim review of the product and suggest changes, if required.

“Every release should be as small as possible, containing the most valuable business requirements.” [Beck, Kent 2000]

Customer Tests: There are a number of customer-defined tests listed during the “whole team meeting” phase. The purpose of these tests is to validate each release to the customer.

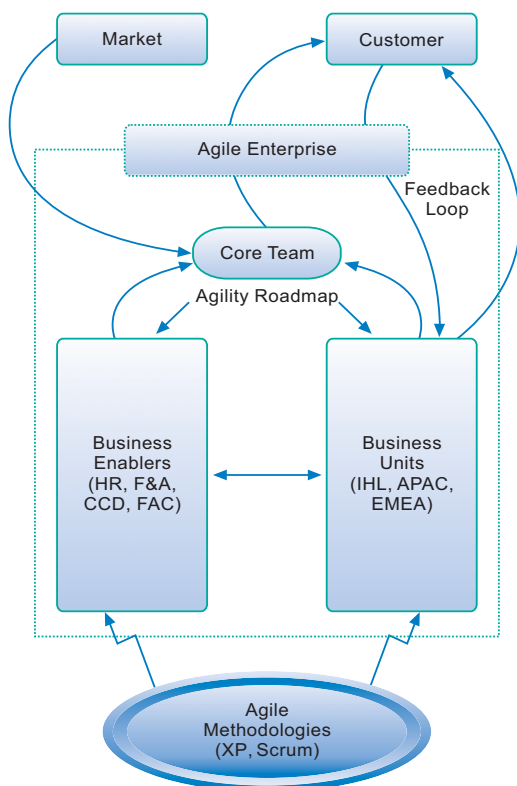


Figure 5. A Generalized Approach to Agility
Source: Infosys Research

Simple Design: A design is simple if it is focused only on the current functionality and doesn't consider potential future functionalities.

"If you believe that the future is uncertain, and you believe that you can cheaply change your mind, then putting in functionality on speculation is crazy. Put in what you need when you need it." [Beck, Kent 2000]

Pair Programming: Pair programming involves two people working together on development of a single program. This leads to prevention and early detection of defects by peer collaboration.

"Pair programming is a dialogue between two people trying to simultaneously

program and understand how to program better." [Beck, Kent 2000]

Test-Driven Development: This step involves developing unit tests in short cycles. At the time of release, all these tests are run to check program reliability and each of these tests must pass for a successful release.

Refactoring: A successful iterative development process has a good design as the foundation. In XP, the process of continuous design improvement is called Refactoring.

"The Refactoring process focuses on removal of duplication (a sure sign of poor design), and on increasing the "cohesion" of the code while lowering "coupling". [Jeffries, Ron 2001]

Continuous Integration: XP stresses on the integration of the units as soon as they are developed. Integration is never left as the *last task*. Late integration invariably complicates the process and increases defect injection rate.

SCRUM

The Scrum methodology gets its name from the huddle formed by rugby players to clash with the players from opposition. It was developed by Ken Schwaber and Jeff Sutherland. The primary thought behind this methodology is that the world is totally unpredictable and hence, it is impossible to accurately plan for the future.

"Scrum relies on self-commitment, self-organization, and emergence rather than authoritarian measures." [Schwaber, Ken 1996]

The Scrum methodology revolves around short 30-day iteration called a Sprint (Figure 4). The Scrum framework is divided into three stages:

Pre-Sprint: The Pre-Sprint stage involves Sprint planning. This is a process of creating a list of features to be incorporated in the system. The owner determines which feature is to be taken up in the next Sprint. A Sprint Goal is also

established which provides a purpose to the team to achieve.

Sprint: Sprint stage leads to the development of the software. The feature picked up from the list is implemented in a 30-day cycle. There is a daily Scrum meeting which improves the visibility of each person's work. Changes to any feature during a Sprint are not allowed, except under extraordinary circumstances.

Post-Sprint: This stage involves customer demonstration, progress review and technical review. This stage ensures that the customer and the team have an early preview to the system.

At the end of this stage, the entire Scrum process is repeated.

AGILITY - A GENERALIZED APPROACH

While there are factors and methodologies contributing to agility, a generalized methodology can also be adopted to achieve agility (Figure 5).

A few of the major participant factors in this approach are:

Core Team: The Core Team takes inputs from external factors such as market conditions, and customers etc, to formulate an "Agility Roadmap". This team comprises of selected people from all units.

Agility Roadmap: The Agility Roadmap is a framework set by the Core Team for the various units of the enterprise. It consists of a set of guidelines which aim to assist the enterprise in achieving the ultimate goal – Agility. The important point is that the guidelines are not rigid and can be changed by the target units.

Business Units (BUs): The Enterprise is divided into various Business Units that have a certain degree of autonomy. This division could be on the lines of geography or types of activities. A BU can be thought of as a "mini enterprise"

within an enterprise. An Agile BU, on its part, forms Core Teams at BU level that are responsible for finding ways to maneuver on the Road to Agility set by the Enterprise-level Core Team. It promotes the use of Agile Methodologies (XP, or Scrum) in all its processes and increases interaction with customers. An agile BU also improves or changes the Agility Roadmap and informs the Enterprise-level core team of the changes.

Business Enablers: The Business Enablers are the units that help in smooth operations of the BUs. Human Resources (HR), Finance & Accounts (F&A), and so on fall into this category. The Agility Roadmap sets certain goals for the Business Enablers which complements the goals set for the BUs.

Customer: Customer in this approach should be one who is interested and involved in the product development. Customer interaction at every level is an important aspect of an Agile Enterprise. This not only gives higher visibility of the process to the customer, but also reduces the inherent unpredictability of the planning and development process. However, it should be noted that the onus is on the Enterprise to appraise the customer of the progress and get his feedback.

Feedback Loop: The Feedback Loop ensures that communication channel between Customers & BUs, Core Team & Business Enablers and Core Team & BUs is always open.

Agile Methodologies: The functioning of BUs and Business Enabler units are aided by the Agile Methodologies such as XP and Scrum. The units decide which methodology would best fit their goals on the Agility Roadmap.

AGILITY IN ACTION – A CASE STUDY

Consider the case of a mid-size CMM Level 5 IT solutions company that has bagged an important contract from a telecommunication major.

The Background: A Senior Project Manager of the IT company is given the overall charge of the assignment. The project is very important as it could propel the company in to a big league.

The Problem: The task was to port the billing systems of the telecommunication company to a new technology platform with some enhancements. The existing billing system was developed on an old proprietary technology, which will cease to be supported in the near future. New regulations have also come into effect that require certain changes to be made to the system. Both the factors make adherence to

disarray. The requirement collection team is yet to finalize the requirements and the training plan was behind schedule and suddenly compounding the situation was a dearth of trained people in the new technology. While initiating corrective measures on the resources front, the senior project manager dispatched a manager onsite to gauge the situation there.

The Analysis: A thorough study of the requirements gathering methods brought some startling facts to light. The requirements collection team didn't have a single point of contact with the client. They were contacted by

Examining requirements gathering methods will indicate that most of the time is being utilized in documentation rather than development!

the timelines sacrosanct and strict penalties are proposed in case of any deviation. The staffing requirements are intimated as soon as the proposal is accepted and the requirements collection phase is started in an earnest. It was noticed later that a major error in resource estimation had crept in.

Planned Quick Fix: A project that is bound by strict timelines simply cannot afford to be short on trained manpower. The project manager, therefore, took a two-pronged approach to tackle this issue:

- Recruit trained technical experts
- Train the internal resources in tandem with the requirement collection phase.

The Crisis: But the project went off the track as the scheduling plan seemed to be in complete

multiple people with new ideas on a regular basis and were swamped with changing requirements and new requests. Hence, most of the time was utilized in documentation rather than the actual work.

The Solution: After a careful consideration of the situation, a few changes are initiated to instill agility and solve the crisis. These changes are aimed not only at solving the issue at hand but preventing recurrence of such issues in the future.

The Training department is given a directive to formulate a policy of in-house training on latest technologies. The recruitment policy is modified as it needs to be driven by certain degree of foresight. If there is a shortfall of experts on a specific technology, a suitable

number should be recruited from outside. These steps would make sure that there is no resource crunch.

The requirements collection process is streamlined. The client is requested to fix a single point of contact for requirement changes and new requirements. The projects which have frequent requirement changes need to do away with traditional steps of Software Development Life Cycle. SCRUM methodology can be used in such projects to prioritize requirements.

A part of the requirements will be frozen for a month. After a month, the client would get a working prototype of the earlier frozen requirements. Now, another set of requirements would be taken up on priority and this process of iterative development will go on. The priority will be decided by the client. The team is asked to document only the absolutely necessary information. These steps would ensure that priceless time is saved and client would get a preview of the system long before the final implementation.

The learning from this project forces the organization to give a serious thought to Agile Methodologies. A central co-ordination team is formed to rollout an Agile strategy for the future.

The Impact: Two months later, the IT project team successfully delivered two working prototypes with limited functionality to the client. The client is thrilled by the quality of work delivered. The final delivery is now on schedule and Agility seems to have saved the day for the IT company.

AGILITY MYTHS:

As is the case with any emerging trend, there are many misconceptions about Agility. Some common myths are –

■ **Agility is for small or mid-sized enterprises only.**

False: Agility is not a function of the size of an enterprise. It demands grace of an expert ballerina from a tiny mouse as well as a giant elephant.

■ **Agility means “lightweight” process, i.e., unstructured methods with little or no documentation.**

False: “Lightweight” signifies the ability of the process to adapt to changes. Agile methodologies do insist on minimal documentation. The emphasis is on documenting only what is necessary and not on churning out stacks of irrelevant documents.


CONCLUSION

Agility in Enterprises is not a modern fad. This is the defining factor which would shape the businesses of tomorrow.

The need to be innovative has never been felt more acutely. This modern era of cut-throat competition can be best exemplified by Darwin’s “Theory of Natural Selection”, i.e., change or perish. Those who embrace change and innovate are the survivors. Others would just fade away. Therefore, it is prudent for today’s businesses to open their eyes to this reality and start their journey on the Agile Speedway.

REFERENCES

1. The Future Of Competition: Co-creating Unique Value With Customers, C. K. Prahalad and Venkat Ramaswamy, Harvard Business School Publishing, 2004
2. Agile Software Development Ecosystems, Alistair Cockburn and Jim Highsmith, Addison-Wesley, 2004
3. Extreme Programming Explained: Embrace Change, Kent Beck, Addison-Wesley, 2000

4. SCRUM Development Process, Ken Schwaber, Paper presented at OOPSLA'95, referenced from <http://www.controlchaos.com/old-site/scrumwp.htm>, accessed July 2005
 5. Agile Software Development with SCRUM, Ken Schwaber and Mike Beedle, Prentice Hall, 2005
 6. Scrum Log, Jeff Sutherland's Website at <http://jeffsutherland.com/scrum/index.html>, accessed July 2005
 7. What is Extreme Programming?, Ron Jeffries' Website at <http://www.xprogramming.com/xpmag/whatisxp.htm>, November 2001
 8. Creativity is not Innovation, David Skyrme's Website at <http://dev.skyrme.com/updates/u17.htm#creativity>, March 1998 
-

About the Author

NAYAN JYOTI KAR

Nayan Jyoti Kar was a Software Engineer with the Insurance, Healthcare and Life Sciences business unit at Infosys. He was associated with projects which involved modernization of legacy applications.

For information on obtaining additional copies, reprinting or translating articles, and all other correspondence, please contact:

Telephone : 91-80-41173878

Email: SetlabsBriefings@infosys.com

© SETLabs 2006, Infosys Technologies Limited.

Infosys acknowledges the proprietary rights of the trademarks and product names of the other companies mentioned in this issue of SETLabs Briefings. The information provided in this document is intended for the sole use of the recipient and for educational purposes only. Infosys makes no express or implied warranties relating to the information contained in this document or to any derived results obtained by the recipient from the use of the information in the document. Infosys further does not guarantee the sequence, timeliness, accuracy or completeness of the information and will not be liable in any way to the recipient for any delays, inaccuracies, errors in, or omissions of, any of the information or in the transmission thereof, or for any damages arising there from. Opinions and forecasts constitute our judgment at the time of release and are subject to change without notice. This document does not contain information provided to us in confidence by our clients.



POWERED BY INTELLECT
DRIVEN BY VALUES