AGILE BUSINESS INTELLIGENCE

OR HOW TO GIVE MANAGEMENT WHAT THEY NEED WHEN THEY NEED IT

Evan Leybourn
Author | Directing the Agile Organisation
Melbourne, Australia
evan@theagiledirector.com

INTRODUCTION

Business Intelligence is a tricky thing. No matter how much effort is put into the design, planning or architecture of a business intelligence project, they rarely go to plan. This is where agile, as a product delivery approach that is responsive to change, comes in. Agile is a generic term that describes over 50 methods and frameworks for working in an adaptable, customer focused, and incremental way. In the context of Business Intelligence, these frameworks cover the full development spectrum; including strategy, planning, design, development, and quality control. Well-known Agile frameworks and techniques include Scrum, Extreme Programming (XP), Test-Driven Development (TDD), and Kanban.

SOME BACKGROUND TO AGILE

Firstly, it is important to understand that Agile is a value system – not a process. Developed in 2001, the Agile Manifesto[1] consists of 4 core values and 12 principles and forms the basis of all agile frameworks.

1. We value individuals and interactions over processes and tools
2. We value working software over comprehensive documentation
3. We value customer collaboration over contract negotiation
4. We value responding to change over following a plan

The values on the right (processes, documentation, contracts and plans) are still important to successful delivery; however, to be Agile, a greater appreciation of the values on the left (individuals, working software, customer collaboration, and responding to change) is needed.

Supporting the 4 core values, are the 12 principles that define the agile mindset. These are the key business attributes that are most important to Agile Business Intelligence teams.

1 Agile Manifesto, Beedle, et al: http://agilemanifesto.org/

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time-scale.
4. Business people and developers must work together daily.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity, the art of maximising the amount of work not done, is essential.
11. The best architectures, requirements, and designs emerge from self-organising teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Understanding these 12 principles is critical to understanding Agile, and Agile Business Intelligence.

AGILE FRAMEWORKS FOR BUSINESS INTELLIGENCE

Regardless of the framework, Agile Business Intelligence projects or teams use Just-In-Time planning, and an incremental, or iterative, delivery process, which allows for rapid change when scope and circumstances change. Customers work alongside the team to shape and direct the outcomes, while the Team regularly delivers partial, though functional, products to the Customer. The product itself will continue to evolve, as each Iteration builds upon the last.

The customer's responsibilities also change within an Agile Business Intelligence project or team, to take on direct responsibility for the delivery of their requirements. Teams and customers work closely together, collaborating towards the desired outcomes.

Depending on the context, and organisational requirements, Agile Business Intelligence can utilise any number of, complimentary, agile frameworks. Common frameworks include Scrum (for its incremental product development processes), Kanban (for its continuous workflow management process), Test-Driven Development (as a mechanism to embed quality control in the work cycle), and Extreme Programming (to provide sustainable delivery within teams). Below is a brief background to each of these frameworks.

Scrum[2]**:** Primarily used as a project management and product development framework, Scrum describes a framework (as shown in Figure 1) for the incremental delivery of complex products. The Scrum framework is primarily team based, and defines associated roles, events, artefacts and rules.

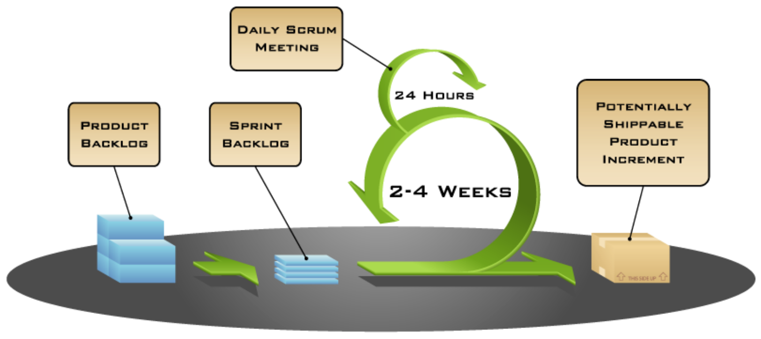

Figure 1: Scrum framework – Mountain Goat Software (CC-AT)

Kanban[3]**:** Kanban (カンバン), which approximately translates as 'signboard', is described as a 'visual process management system that tells what to produce, when to produce it, and how much to produce'. At its simplest, each prioritised Task (or Card) on a Kanban Board passes through a visualisation of the Team's process, or workflow, as they happen. Each primary activity in the Team's workflow is visualised as columns on the Kanban Board, usually starting at Task definition, and finishing with delivery to the Customer.

Test-Driven Development[4]**:** Test-Driven Development (TDD) is primarily a software engineering process that forces programmers to write small, incremental verification tests prior to writing each function of the software. Each set of verification tests defines the outcome of a single feature or improvement. This 'test first' approach encourages simple design, concise development, and confidence in the product.

Extreme Programming[5]**:** Extreme Programming (XP) is an Agile development framework which, like all Agile frameworks, advocates incremental delivery and responding to changing customer requirements. XP's focus is on the method and role of the delivery team, and defines basic activities within the software development process.

There are other development concepts, not directly associated with agile, that should also be considered to ensure the success of a business intelligence project. These include Unit Testing, Continuous Integration or Code Generation.

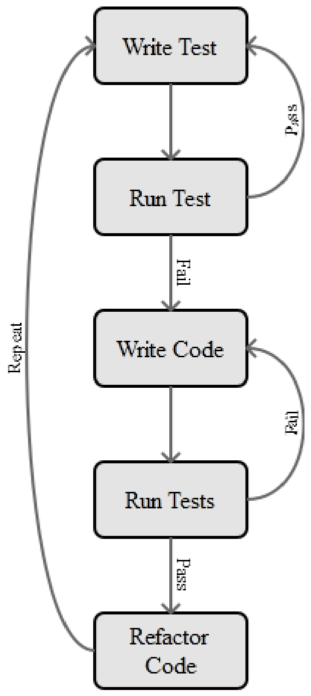

Figure 2: Test-Driven Development flowchart

2 Scrum Guide, Schwaber and Sutherland (2011).

3 Kanban: Successful Evolutionary Change for Your Technology Business, Anderson (2010).

4 Test-Driven Development by Example, Beck (2003).

5 Extreme Programming Explained: Embrace Change, Beck (1999).

BUSINESS INTELLIGENCE PLANNING AND DESIGN

Prior to beginning any Agile Business Intelligence project, and to minimise later rework, an initial product backlog, or requirements backlog, needs to be defined by the customer. The product backlog describes the current and best-understood requirements, sometimes known as user stories. The highest priority requirements, those that will be delivered next, are then expanded with additional detail. This ensures that the outcomes at the end of an iteration are always usable, and deliver value to the customer.

To contain the impact of change, each business intelligence user story is initially defined at a high level. The level of detail required should contain sufficient information to help prioritise and identify problem areas for prior to beginning the process, without wasting effort. For example:

1. What quantity of data is being dealt with? - e.g. a CMS with 1,000 records or a financial system with 100,000,000 records
2. How quickly does the data grow? - A CMS which adds 100 new pages a day will be easier to manage than a financial management system which add 1,000,000 records
3. How frequently does this data change? - Compare the financial system in which the data never changes (and so there is no need to manage data revisions) to a CRM in which the data changes daily
4. When does this data change? - We can optimise the ETL process if we know the data is updated at the end of the month, rather than try and ETL the same data every day.
5. How much of this data is duplicated elsewhere? - If the data is duplicated in another system, and the other system is authoritative, we can simplify the ETL process.
6. How much of this data is obsolete or irrelevant? - This comes down to GIGO. If the data is obsolete or not relevant to current decision making, including it in the data warehouse would be counter productive
7. How is this data used? - This question is used to inform the necessity/benefit of the data and should be used as part of the cost-benefit analysis.
8. What reports currently use this data, and are they satisfactory? - Often the transactional system (e.g. finance system) provides its own reports. If these are satisfactory and do not need to strength of a BI system behind it, it de-prioritises the data source for extraction.
9. What access is available to the data, and how can we extract it? - This is a purely technical question that informs the effort involved in the ETL process. Is it a database, can we connect to it directly, via ODBC, do we need software engineers to write extraction scripts etc?

AGILE TEAMS

The key distinction between Agile Business Intelligence and traditional business intelligence teams is the use of a cross-functional team structure. A cross-function team is one where individuals with different, but complementary, skills, work together as a team and are empowered with personal authority and accountability. Each team should contain all the key skills required to deliver on the business intelligence requirements. Each cross-functional agile team is typically between 5-9 full-time staff, where the whole team works towards a single, specific outcome.

Benefits to this integration include faster delivery times, rapid response to new issues, and improved information sharing across the organisation. However, to realise these benefits requires that all team members are committed to the outcome, and adaptable in their role. This is different to the traditional hierarchical or matrix management structures, where one team would start the process, and at pre-determined stages, request input from, or handover to, another team. By passing work between silos, strict matrix structures lack consistent ownership of work, cause poor communication between departments, and increase delays in the overall process.

Each cross-functional team also consists of both technical and business people. By utilising business and technical experts working together, outcomes can generally be met sooner and more accurately than if the teams were separate. Roles may include:

- Business Analysts - the interface between the BI teams and external business teams
- Business Specialists - often a subset of the business analyst, business specialists (or subject matter experts) are sourced from the business teams responsible for creating or consuming the organisational data, e.g. call centre staff, program / policy staff, or retail staff
- Report / Technical Writers - only required where high quality and complex reports for external business teams is needed. It is also common for the BI team to provide a “self-serve” tool
- Statisticians / Demographer - only required if when dealing with complex data and there is a need to do in-depth statistical analysis
- BI / ETL Specialists - specialist team members who understand the organisational BI and ETL tools, be it Pentaho, Cognos, Oracle BIEE, etc
- Data Analysts - responsible for reverse engineering data sources, maintaining the databases and cubes and interpreting business need into information required
- Systems / Database Administrators - responsible for maintaining the BI servers and DBMS'

Please note that these are roles and responsibilities, not necessarily individual people. e.g. an ETL specialist and data analyst can be the same person, as can a data analyst and DBA. As for organisational structure, an Agile Business Intelligence team would ideally report to a business manager, rather than a technical manager or CTO. The technical manager will generally be focused on the systems and technology, whereas the business manager will be focused on the information outcomes.

TEST-DRIVEN DEVELOPMENT

Formal quality control and testing frameworks, such as Test-Driven Development, are a critical component of Agile Business Intelligence. The consequence of poor testing is relatively well understood and includes; inefficient design, significant data integrity and validation issues, and defects within ETL and reporting scripts.

Dedicated testers are embedded within each team to undertake quality control while still ensuring a separation between the doers and testers. One dedicated tester per team (between 5-9 people) should be able run complete quality control tests at roughly the same rate as requirements are completed. High-risk or high-complexity requirements may require additional testers per team (or smaller teams); play with the team numbers until equilibrium is reached. That is, where everyone is busy, but no one is being a bottleneck.

Test-driven development (TDD) aims to improve business intelligence quality by focusing on early and regular testing. The core premise is that automated or “unit” tests are written before each software component. Therefore, the development process should look something like this:

1. Write the (failing) test
2. Run all tests (ensuring that the new test fails)
3. Write the code
4. Run all tests
5. Repeat steps 3-4 until the test passes
6. Refactor the new code to acceptable standards

To make this work within a BI context, it is important to define user stories as discrete components. The first step is to define our test scope, which at its simplest is an ETL script that, given a known input, will output data that is reasonable and within expected norms. This scope can include:

- The number of results - e.g. we know there are 1,000 records in the data source which change daily, so our ETL should extract approximately 1000 records each day
- The scale of results - e.g. financial transactions must be between $1 and $1000
- The relationship of results - e.g. the average or standard deviation should be similar each day
- Logical tests - e.g. the sum of all transactions for a customer should equal the customers balance
- Known good tests - e.g. customer 123 exists and has a known form
- Relationship tests - e.g. that each transaction has an associated customer

Under the TDD framework, these tests should be written before any ETL scripts (though usually after the model/schema has been designed). It has been shown that this encourages developers to create simpler designs and inspires confidence in their work.

To get the greatest benefit out of TDD and to ensure a robust testing process, all tests should be setup and run through good continuous integration environment. This will automatically run the tests on a scheduled (or triggered) basis and alert the BI administrators of any failing tests, which can forewarn of issues in the data sources and data warehouse.

RISKS

As with everything, Agile Business Intelligence teams and projects are not without their risks. Indeed, while Agile brings many benefits, the impact of some risks are increased. Some risks to be aware of include:

- Lack of customer engagement: Where projects stalled while waiting for clarification or decisions from the customer
- Competing priorities: Where assigned team members also have conflicting BAU responsibilities
- Low team morale: Low morale (for example, due to lack of delivery or a low understanding of project value) can affect team productivity and, in turn, can lead to even worse morale
- Insufficient allocated time for delivery: Where more work has been allocated than can be delivered in a given deadline. It can be helpful to visualise a project teams as a pipeline. The width of the pipe is the team size and the length is the time available to deliver. If an estimate is incorrect or a new requirement comes into the pipe, the lowest priority requirement will fall out the end. In Agile terms, the velocity of each team doesn't change just because they are given more work
- Unfocused development: Where developers go off on tangents building "frameworks" that add no direct value for the customer
- Poor quality (or inconsistent) test data: Poor test data causes inaccurate tests results (both false positives and negatives) resulting in rework

SUCCESS MEASURES

Finally, it is important to measure the success of an Agile Business Intelligence project. A set of specific success criteria, with measurable targets, need to be defined to validate when the Agile Business Intelligence goals have been achieved. Success criteria should be concise, realistic, directly measurable and include both quantitative measures, based on facts & figures, and qualitative measures, based on feedback & opinion. Agile specific maturity measures include:

- Staff are trained & experienced in Agile Business Intelligence and associated methods.
- Staff have an understanding of the underlying reasons for moving to Agile Business Intelligence
- Staff are directly empowered to engage with & deliver to the customers
- Staff are conversant in the work, QA, and release procedures
- Customers have been trained in their new responsibilities
- Customers (or their representatives) are involved in the teams daily activities
- Customers actively define & prioritise requirements at least once per Iteration
- Clearly defined business processes exist for all domains

As a final note, not all BI projects are suitable for an agile approach. Projects and teams with low morale, no staff or executive buy-in, high staff turnover, or a lack of trust between themselves & the customer, need to address these issues before beginning any Agile Business Intelligence project.