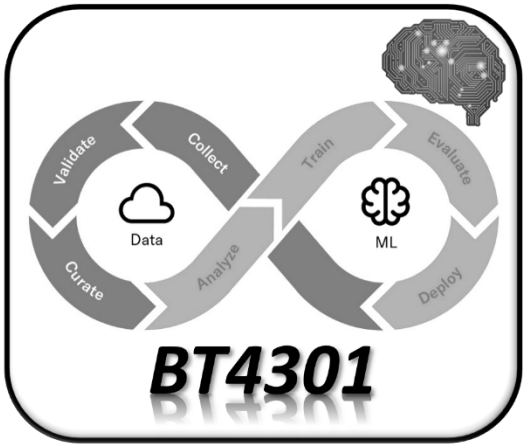


Lecture 2

Agile Methods and Analytics

BT4301 – Business Analytics Solutions Development and Deployment
AY 2023/24 Semester 2

Lecturer: A/P TAN Wee Kek

Email: tanwk@comp.nus.edu.sg :: **Tel**: 6516 6731 :: **Office**: COM3-02-35

Consultation: Wednesday, 9:30 pm to 10:30 pm. Additional consultations by appointment are welcome.

Learning Objectives

- At the end of this lecture, you should understand:
  - Agile methods.
  - Integrating agile methods with analytics.

Readings

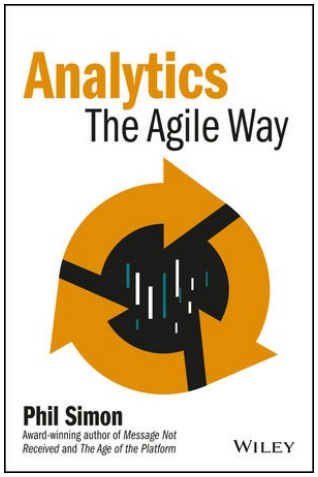
- Reference Book 2:

  Analytics: The Agile Way

  Author: Phil Simon

  1st Edition (2017), Wiley

  https://linc.nus.edu.sg/record=b3751448



Readings (cont.)

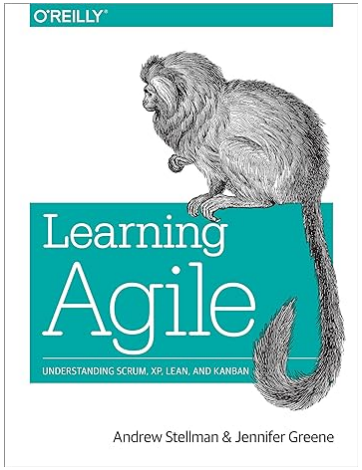
- Reference Book 3:

  Learning Agile: Understanding Scrum, XP, Lean, and Kanban

  Authors: Andrew Stellman and Jennifer Greene

  1st Edition (2014), O'Reilly Media

  NLB Link



Readings

- Required readings:
  - Reference Book 3 – Chapter 1 and 2
  - Reference Book 2 – Chapter 6
- Suggested readings:
  - Reference Book 3 – Chapter 3

Origin of Agile

- The **agile** paradigm originates from IT software development.
- **Agile** promises projects that:
  - Complete on time.
  - Deliver high-quality software.
  - Build on well constructed and highly maintainable code.
  - Make their users happy.
  - Involve developers who work during normal hours and spend their nights and weekends with families and friends :)

XP is more focused on engineering & coding practices.
Scrum is more high-level.

Origin of Agile (cont.)

- The two most popular agile methodologies are:
  - Scrum
  - Extreme Programming (XP)
- Scrum will be discussed in a future lecture together with Kanban.

Factor	Scrum	XP
Engineering Practices	No	Yes
Project Management Practices	Yes	No
Accept Changes in Iteration at Any Time	No	Yes
Requirement Prioritization	No	Yes
Refactoring	No	Yes
Pair Programming	No	Yes

Origin of Agile (cont.)

Factor	Scrum	XP
Project Size	Medium to Large	Small to Medium
Self-organization	Yes	No
Test Driven Development	No	Yes
Unit Testing	No	Yes
Design Approach	Design Centered	Code Centered
Documentation Level	More	Less
Team Size	< 10 and Multiple Teams	< 10
Code Style	Not Specified	Clean and Simple
Technology Environment	Not Specified	Quick Feedback
Physical Environment	Not Specified	Co-located and Limited Distribution
Business Culture	Not Specified	Collaborative and Cooperative

does not care about coding practices

Back to Basics – What is Agile?

- Agile is a set of methods and methodologies that helps a team to:
  - Think more effectively,
  - Work more efficiently.
  - Make better decisions.
- Agile methods and methodologies address all areas of software engineering:
  - Project management.
  - Software design and architecture.
  - Process improvement.

Back to Basics – What is Agile? (cont.)

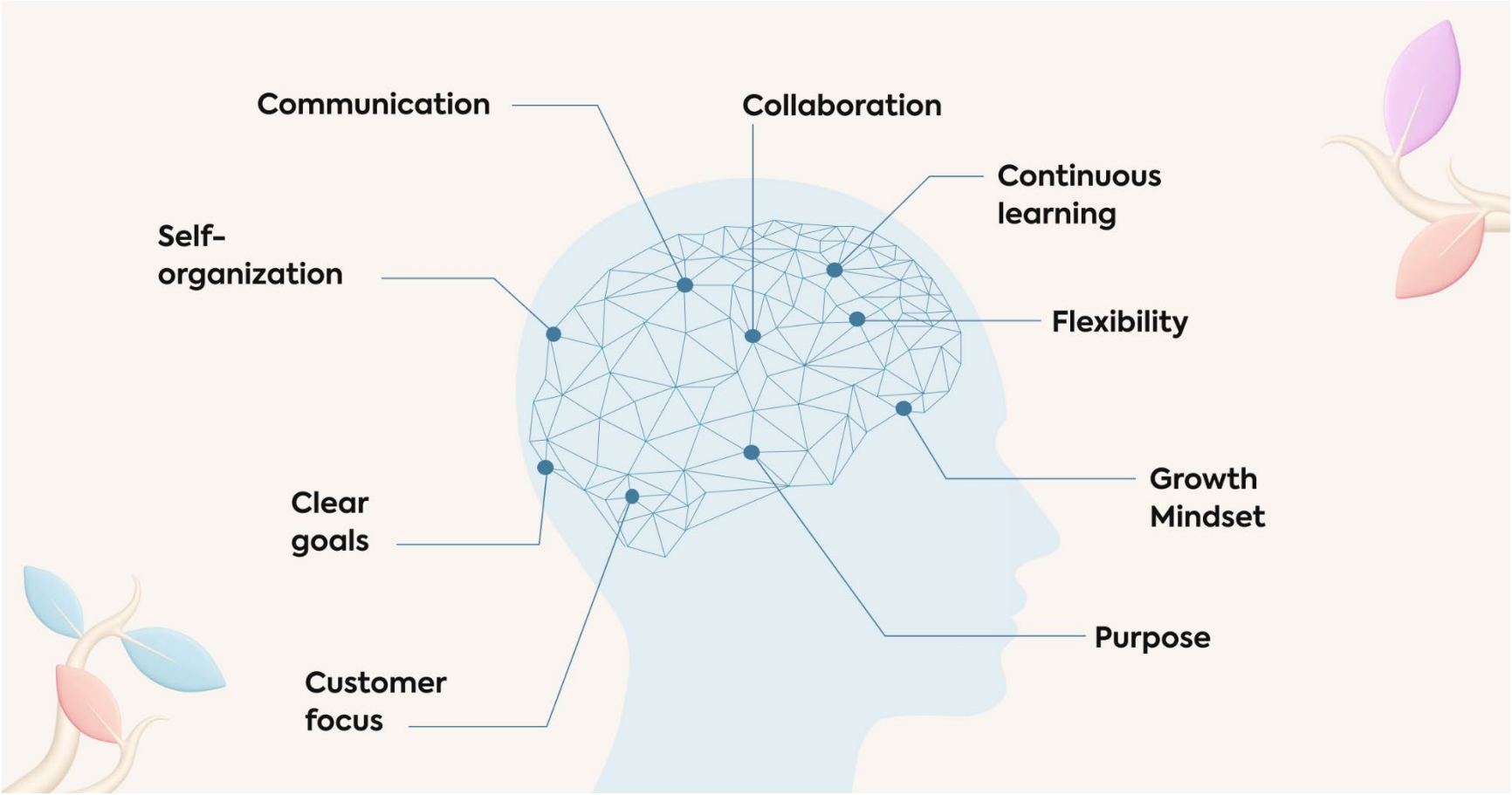
- Agile is also a mindset:
  - Help team members share information so that decisions are made collectively and not by an individual manager.
  - Adopt a right attitude,
  - E.g., a common agile practice is daily standup:
    - Team members must be excited about meeting each other.
    - Open up their daily work to scrutiny.
    - Project manager genuinely listen to the concerns of team members.
    - Effective daily standup requires everyone to have the right mindset to work faster, communicate directly and get things done easily.

Very important!

In the groups, we will need to do a daily standup in front of the class.

And have a positive mindset!!!

Back to Basics – What is Agile? (cont.)

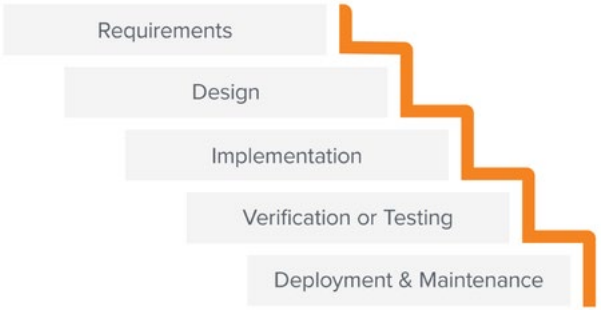


Waterfall assumes that it is possible to write all requirements upfront and that requirements will not change over time.

Ineffective Waterfall Process

- In a **waterfall** process:
  - The analysis team tries to write down as early as possible a complete description of the software that will be built.
  - Once all stakeholders agree on exactly what the software must do, the requirements specification is then handed over to a development team.
  - The development team will build exactly what is written.
  - Then a testing team comes in afterward and verifies that the software matches the requirements specification.
- This is known as "Big Requirements Up Front" (BRUF).

Just like a construction worker. Developers can not express their opinions, nor highlighting problems in the requirements.



Ineffective Waterfall Process (cont.)

- The assumption is that a perfect requirements specification can be created.
- But in real-world context, things are ever changing:
  - Requirements specification could become inaccurate by the time it reaches the development team.
  - The software becomes disastrously wrong by the time it is completed.
- Waterfall process typically fails due to:
  - Overly rigid documentation.
  - Poor communication.
  - Bugs – Mainly arising from major changes made to the code.
    ↳Bugs are more costly to solve at the very end.

Ineffective Waterfall Process (cont.)

- Some waterfall projects are successful due to stable requirements:
  - Requirements are right at the beginning and needed very few changes.
  - E.g., medical software. → very stable regulations. Requirements do not change.
- But there are other common characteristics of successful waterfall projects:
  - Good communication – Constant communication between users, managers and executives throughout project.
  - Good practices – Code review, automated testing, etc.
- So, effective waterfall projects resemble agile values, principles and practices.

Waterfall Case Study – Slot Machine Project

- **"Slog-o-matic Weekend Warrior"**
- **The team**:
  - Dan – Lead developer and architect
  - Bruce – Team lead
  - (Old) Project manager
  - Tom – Product owner
- **Initiation phase**:
  - Bruce, Dan and Tom spent first few weeks of the project building up the specification for the new slot machine.
  - Tom was only in the office half the time. → Product owner was actually the key person!
  - Once the three had agreed on the requirements, a big meeting was arranged with the CEO and senior managers to review the requirements, make changes, and get approval to start.

Waterfall Case Study – Slot Machine Project (cont.)

- **Development phase**:
  - Tom went out of office again leaving the work to Bruce and Dan. Not good communication between technical & business team
  - Bruce and Dan broke the project down into tasks, dividing them up among the team to build the software.
- **Ending phase**:
  - When the team was almost done building the software, Tom gathered the stakeholders into a big conference room for a demo of the nearly completed software.
  - CEO dropped a bombshell on "video poker mode":
    - Under the impression that the software could be deployed to slot machine hardware or video poker hardware.

Waterfall Case Study – Slot Machine Project (cont.)

- There was a lot of discussion of this between the senior managers, the board, and the owners of their two biggest customers.
- **But nobody had bothered to tell the team!** Developers were the last to know!

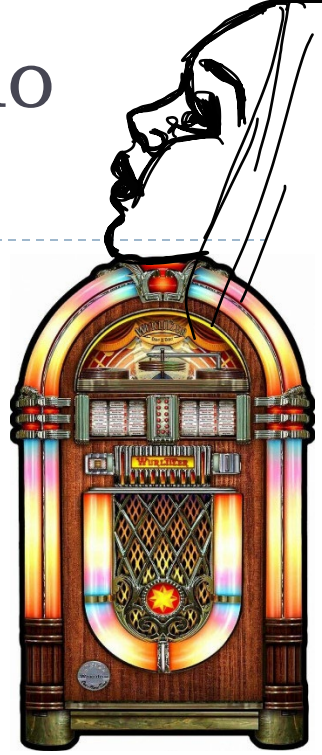
- **Aftermath**:
  - Bruce and Dan had to remove enormous amount of code previously written and replace them with code retrofitted from the video poker project.
  - Spent lots of time troubleshooting weird bugs caused by integration problems.
  - Also increased difficulty of maintaining the code.

Waterfall Case Study – Slot Machine Project (cont.)

- **Consequences**:
  - Project manager left the company and replaced by Joanna.
  - Tom had to face the customers when they run into problems.
  - Biggest customer did not cancel the contract but awarded the new big contract to a competitor.

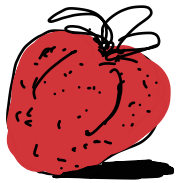
However, Waterfall is very used in contractor relationship, where a job has fixed requirements and the project is paid on a feature-basis

Agile Case Study – Streaming Audio Jukebox Project



- **The team**:
  - Dan – Lead developer and architect
  - Bruce – Team lead
  - Joanna – Newly hired project manager
  - Tom – Product owner
- Joanna suggested that it was time to go **agile**.
- Discussed what "agile" really meant to each of them:
  - Bruce – Agile development.
  - Joanna – Ability for a project to handle **changes**.
  - Dan – Zero documentation and jumping straight into code.

Agile Case Study – Streaming Audio Jukebox Project (cont.)

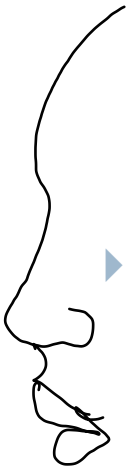


- Tom:
  - Had no idea what the other three were talking about.
  - But was happy that they would be giving him lots of demos along the way.
  - He could avoid what happened last time – Major change discovered only during ending phase.
- Team members started educating themselves about agile techniques, practices and ideas.
- **Towards an agile process**:
  - Dan:
    - Started unit testing, test-driven development, automated build script, build server ➔ **DevOps**
    - Saw improvement in code quality.

able to identify problems early on.

Agile Case Study – Streaming Audio Jukebox Project (cont.)

- Joanna:
  - Attended Scrum training and became the Scrum Master.
  - Broke the project down into iterations (not tasks).
  - Track project progress on task board, project velocity and burndown charts, etc.
- Tom:
  - Became the product owner.
  - Write user stories for the team.
  - Build release plans based on the stories.
  - Felt he had direct control over the software deliverable.
- Bruce:
  - Started holding daily standup meetings with the team.
  - Tom attended the daily standup meetings too.



Because it is done in an incremental way.

Agile Case Study – Streaming Audio Jukebox Project (cont.)

- Held retrospectives as a team at the end of each iteration.

- **"Better-than-not-doing-it" results**:
  - It all worked.
  - The team improved, and the project got better ... up to a point.

Agile Case Study – Reflections

- **Has the team really become agile?**
- They had adopted a lot of great practices.
- There were improvement and happiness but also hesitation:
  - Dan: Dan feels he could be doing a better job.
    - Found himself making some technical sacrifices to meet the schedule.
  - Joanna:
    - Breaking the project down into short iterations made her felt a bit blind.
    - Increasingly dependent on the team to tell her what's going on at the daily standups.
    - Overly focused on reacting rather than planning.

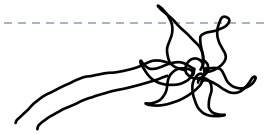
she is losing control of the process, because decisions are now shared.

Agile Case Study – Reflections (cont.)

- Tom:
  - Felt that he was expected to work for the team full time.
  - Felt that the team was pushing all of the responsibility for building a great product back on him, and he didn't have all the answers. Because he is not a tech guy.
- Bruce:
  - Something seems unsatisfying and incomplete, and he was not sure why.
  - Bruce liked the agile adoption – Made things better, cut down on the personal heroics, and reduced the number of long nights and weekends of work. → He missed being the hero?
  - But he also felt that agile brought its own set of problems.

Agile Case Study – Reflections (cont.)

- **A fractured perspective**:



  - Many teams have discovered that agile practices can help them mitigate the waterfall problems but its not as straightforward.
  - Seeing only the tools, techniques, and practices is just the first step in "going agile," and it has a problematic side effect.
  - Everyone has a different view of agile practice. → very common issue. This causes conflict.
- Example – Agile practice of writing user stories:
  - A user story is a way to express one very specific need that a user has and can be written in a strict structure or flexible format.
  - Each team member sees user stories differently.
  - In waterfall, Dan had a detailed spec that left little room for flexibility.

Agile Case Study – Reflections (cont.)

- In agile, Dan had the freedom to make broader decisions about what he built.
- That can be a good thing, but it can also lead to some problems.
- "As a bar patron, I want to be able to play the newest hit that was just released today"
  - Dan thought this meant allowing patron to play any hit as soon as it was uploaded to the server.
  - But Tom was upset as bar owners had to pay higher royalty fees.
  - Tom would prefer a feature that allowed patrons to play latest hits just enough to be happy but not ran up excessive costs.

There is a conflict between the tech and business team.

The Agile Paradox

- Teams that are new to agile sometime run into trouble:
  - They have not really changed from their old, waterfall-like ways.
  - Just adopting agile practices is not sufficient to break them out of the problems that cause conflict and avoidable changes.
  - A team that adopts agile practices one at a time will often only get "better-than-not-doing-it results".
- How does a team get past this problem?

A team needs to get these values to really leave Waterfall problems behind.

Agile Core Values

- A.k.a. **Manifesto for Agile Software Development**:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan in a systematic manner.
- **Understanding and effectively working with agile starts with understanding these values.**

Agile Core Values (cont.)

- **Individuals and Interactions Over Processes and Tools**:
  - People can go wrong when they blindly follow a process.
  - A great tool can sometimes help you do the wrong thing faster.
  - More important to understand the people on the team, how they work together, and how each person's work impacts everyone else. As a tech team, you need to understand the business part.
  - Agile teams value individuals and interactions over processes or tools:
    - Encourage practices that support individuals and interactions such as daily standup meetings and retrospectives.
    - User stories – The stories are less important than the fact that they help to create the conversation. User stories are only the means to an end.

Agile Core Values (cont.)

- **Working Software Over Comprehensive Documentation**:
  - There is so much that can be documented in a software project:
    - Difficult to predict what is going to be useful in the future, and what will gather dust.
    - End up documenting everything.
  - Agile teams value working software over comprehensive documentation:
    - Working software is software that adds value to the organization.
    - Documentation is only a mean toward that end.
    - But does not mean zero documentation.

Agile Core Values (cont.)

- Focus on documentation that helps in understanding the problem, communicating it with the users, and correcting problems.
- E.g., wireframes, sequence diagrams, automated test cases, etc.

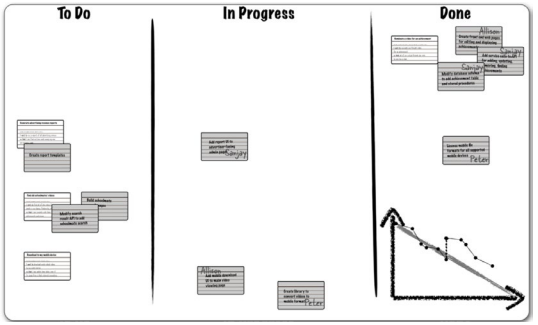
- **Customer Collaboration Over Contract Negotiation**:
  - Different teams within a same company should not adopt a contract mindset, e.g., avoid internal SLAs.
  - Contract mindset is defensive and counterproductive:
    - A developer ends up constantly trying to protect himself.
    - Less able to try new ways to collaborate and innovate with the people who need to use the software being built.

Agile Core Values (cont.)

- Agile teams should have a product owner who is a real, first-class member of the team.

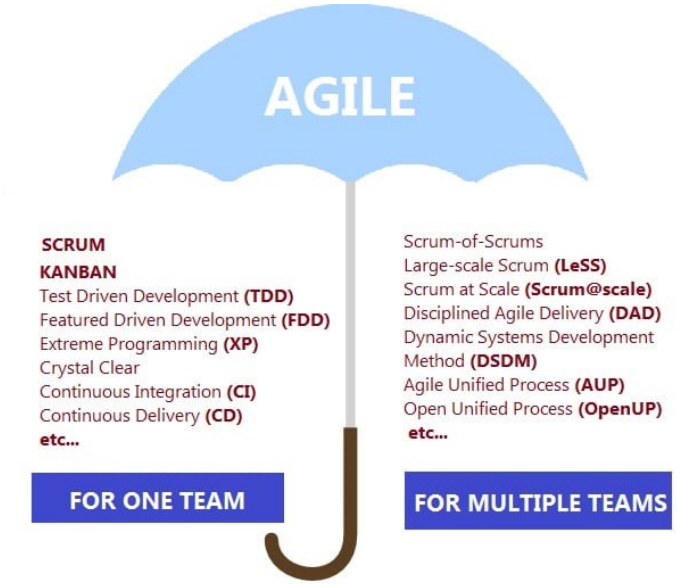
- **Responding to Change Over Following a Plan**:
  - In project management – "plan the work, work the plan" but unfortunately, if you work the wrong plan, you would be building the wrong product.

  - Agile team needs to constantly look for changes, and to make sure that they respond appropriately when there is a change.
  - If the circumstances change, the project needs a new plan.
  - A taskboard can help a team to make right decisions about responding to changes.



Agile Methodologies

- There is a big gap between:
  - Understanding the agile values and principles; and
  - Actually changing the way an agile team functions.
- There are agile methodologies that are intended to help teams adopt agile and improve their projects.
- We will focus our discussion on **Scrum** and **Kanban** in a future lecture.



Agile works when the team is capable. If team members are less experienced, they might not put the freedom in good use.

Agile Analytics

- Agile methods are old hat in the technology sector, but they have since been adapted to other industries:
  - Example of General Electric:
    - GE Digital COO was a software engineer and piloted Scrum in GE.
    - Develop industrial internet applications with Scrum.
    - Gradually applied Scrum to management processes such as operating reviews.
- Although the notion of **agile analytics** is relatively new, it has gained traction fast.
- We will examine a simple and general framework for obtaining analytics insights in an iterative or agile fashion.

A Framework for Agile Analytics

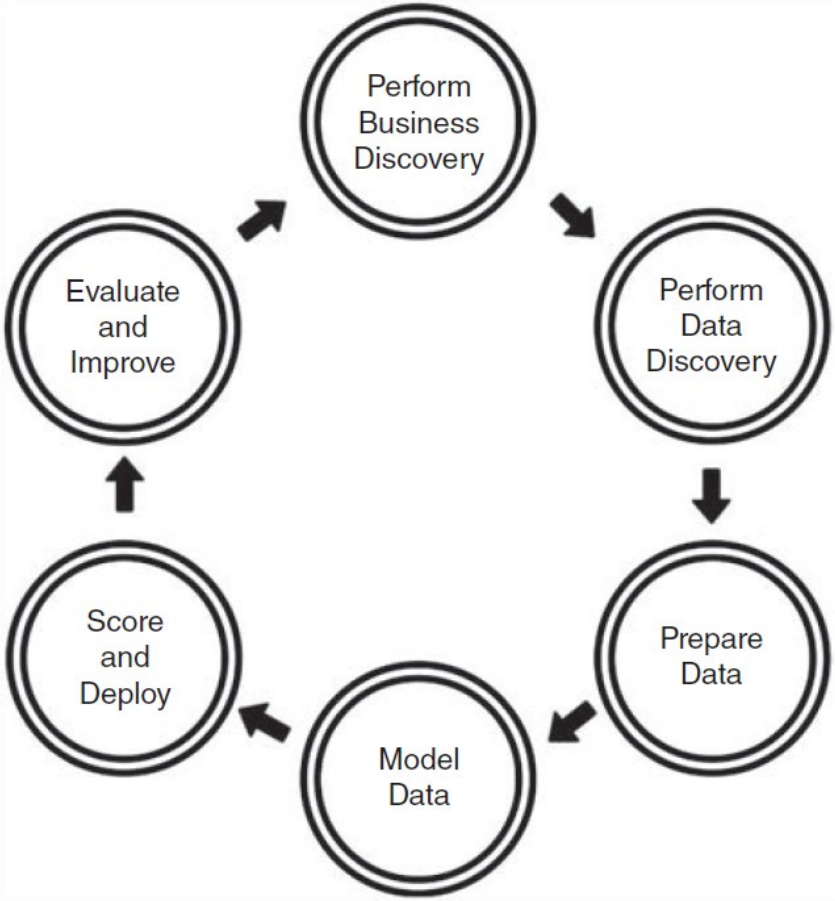


Figure 6.1 A Simple Six-Step Framework for Agile Analytics
Source: Model adapted from Alt-Simmons' book Agile by Design. *Figure created by Phil Simon.*

Perform Business Discovery

- Attempt to solve a real business problem:
  - Why are customers churning?
  - Why can't inventory level be accurately predicted.
- Start by answering key questions:

  - What are you trying to solve?
  - What behaviour(s) are you trying to understand, influence and/or predict.
  - What type of data would you need to address these issues?
  - Is the project even viable?
  - Does your organisation possess the time, budget and resources to undertake the project?
  - Is your organisation committed to the project?

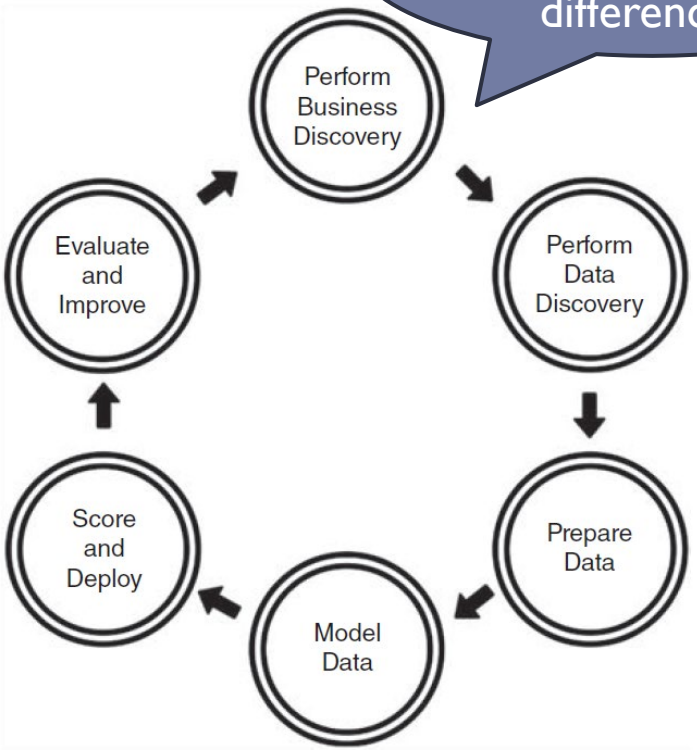
Perform Business Discovery (cont.)

- What happen if you don't answer these questions? What if the project takes longer than expected.

- Not all stakeholder will agree whether a project is viable:
  - Encourage disagreement among stakeholders.
  - Hold discovery workshop or brainstorming sessions to flash out ideas. You vote with the stakeholders.

- Start with a testable hypothesis or working theory:
  - Initial hypothesis – Customers are leaving because your products are too expensive.
  - Null hypothesis – Customers are not leaving because your products are too expensive.

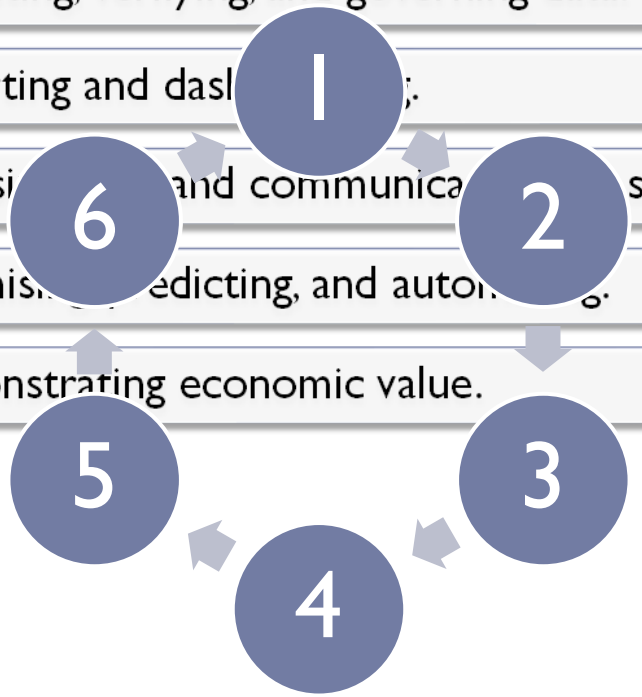
Doesn't This Sound and Look Familiar?



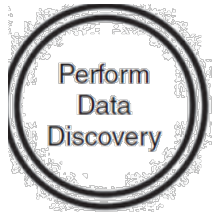
But wait, there some important differences!

Analytics value chain, but as an ongoing process (we can then apply agile).

1. Understanding what to analyse.
2. Collecting, verifying, and governing data.
3. Reporting and dasl[illegible].
4. Analysi[illegible]nd communica[illegible]socialising.
5. Optimis[illegible]edicting, and auto[illegible].
6. Demonstrating economic value.



Perform Data Discovery



- Critical data related questions:
  - Where does the data reside, i.e., data sources?
  - Is the data even available?
  - Is the data legal to use? Is the data free to use?
    - Exercise caution for competitive intelligence data as espionage may be unethical and illegal.
    - Consumer data is protected by privacy law.
  - Are you able to retrieve the data in a clean and usable format or is scaping required?
  - Is use of data restricted? Can you pay to circumvent the restrictions?
  - How old is the data? Does it age well, i.e., still relevant?

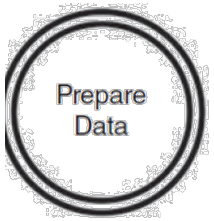
old data might not be usable.

Perform Data Discovery (cont.)

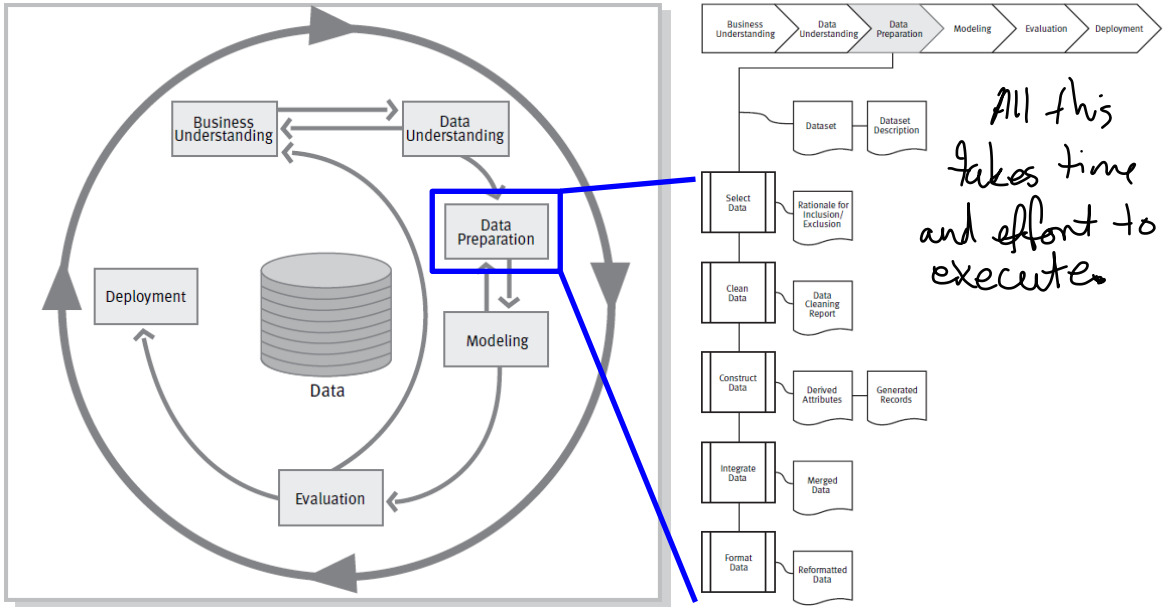
- If the data is from internal sources, who owns the data? Can the owners share the data?
- Is the complete, accurate and deduplicate?
  - Classic problem of number of unique customers or visitors.
- **Be conservative when estimating the efforts, duration and resource required to obtain data.**

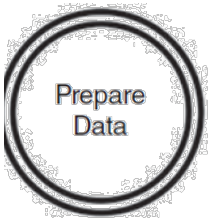
A lot of time, getting data implies a lot of data engineering that was not planed.

Prepare Data



- Data collected likely will contain errors, inconsistencies and missing values.
- Data quality is important and thus data preparation may take up a significant amount of time:
  - Recall that in CRISP-DM, we have also emphasized that data preparation entails many sub-steps.





Prepare Data (cont.)

- Classic example:
  - Expected data:

So, what are the problems?

Customer_ID	PurchDate	PurchAmt	ProductCode
1234	1/1/08	12.99	ABC
1234	1/19/08	14.99	DEF
1234	1/21/08	72.99	XYZ

Source: Phil Simon.

  - Actual data: wide data will have a lot of NULLS

Customer_ID	Purch Date1	Purch Amt1	Product Code1	Purch Date2	Purch Amt2	Product Code2
1234	1/1/08	12.99	ABC	1/19/08	14.99	DEF
1235	1/1/12	72.99	XYZ	1/19/08	14.99	DEF
1236	1/1/08	12.99	ABC	1/19/08	72.99	XYZ

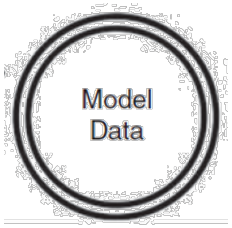
Source: Phil Simon.

Prepare Data (cont.)

- Critical data preparation questions:
  - Who or what generates the data?
  - If people are responsible for generating the data, are they trained to enter it properly? Be very worry if humans are involved.
  - Is the data coming directly from the source system or from another source such as data mart or data warehouse?
  - How is the data currently generated and has that ever changed?
  - How much data is generated?
  - Is the data is flawed or incomplete? What are the downsides?
  - Is certainly data essential to proceed? Are there alternatives?

Prepare Data (cont.)

- How complex is the data?
- How frequently is the data updated?



Model Data

- Goal of a **model** is to understand, describe and/or predict an event.
  - if it is too accurate, the model is probably overfitting
- No model is completely accurate but the important question hinges on whether a model is useful.
- Important points to note about models:
  - Which variables are important?
  - The absolute and relative importance of these variables.
  - Which variables ultimately don't matter.
- Models should be actionable leading to improvement in business outcomes.

We always prefer reasonable good model, but simple.
Models should be explainable.

Model Data (cont.)

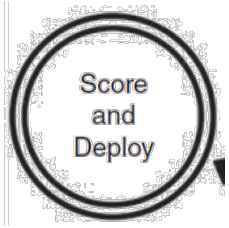
- A simple example:
  - A model to predict which individuals will visit your website.
  - A model to predict which individuals will buy something from your website.
  - Click through rate versus conversion rate?
  - Which model is actionable and which model is more important?

Both are actionable and both are equally important.

- Keep models simple and avoid over complications:
  - Model interpretability and explanability are important.

Because if business people will not trust it!

Score and Deploy



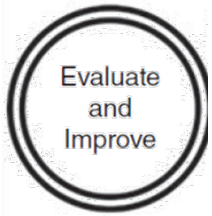
- Forecasting is not a static process:
  - Forecasting attempts to describe and predict ongoing events of far greater complexity.
  - Models need to evolve over time.
  - Customer churn, employee attrition and credit card default rates do not end with a successful model.
- Score and deploy phase starts the process of assessing the viability of the model.
- Important questions to answer:
  - Is your model working well?
  - Are you measuring what you sought to measure?

It is important to have clear measurement definitions

Score and Deploy (cont.)

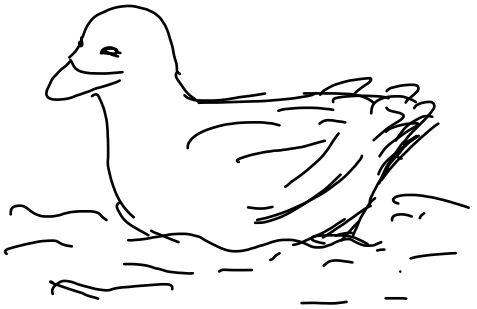
- Even if you are looking at the right (independent) variables, are their weights appropriate?
- How confident are you in your predictions?
- What is an acceptable level of uncertainty given that 100% accuracy is not possible?

Evaluate and Improve



- After developing several iterations of a model:
  - It is important to check if it is describing or predicting what is expected.
  - Audit your model.

- Model updates take one of three forms.
- **Simple data refresh**:
  - Replace a model's existing dataset with a different one.
  - The new dataset may include newer records, older ones or a combination of both.
- **Model update**:
  - Complete or partial rebuild.
  - May entail new variables and associated weights.



Evaluate and Improve (cont.)

- **Combination**:
  - This method is a hybrid of the first two methods.
  - Significantly alter the model and run a different dataset through it.
- Manage expectation of model performance:
  - Absolute versus relative rules.
  - Events that are harder to predict likely has lower model performance.
  - A model for simpler event with better performance may be inadequate.

Evaluate and Improve (cont.)

- Important questions to answer:
  - What data sources are missing? Which ones are worth including?
  - Which data sources may dry up? What should you do if that happens?
  - Which data sources should be retired?
  - Which weights need adjusting? By how much?
  - Will your model improve or diminish if fundamental changes are made?
  - What are the time implications?
  - What happens if you make a big mistakes?

What is Next?

- In the next lecture, we will discuss how to apply this general framework within established agile methodologies:
  - Scrum
  - Kanban

Summary

- The agile paradigm originates from IT software development.
- Agile methods and methodologies have proven to increase the success of software development projects.
- But adoption of agile should be done carefully in accordance with the agile values.
- Agile has been adopted to various other industries and of course to analytics.
- The six-step general framework for agile analytics exhibit the important characteristics of ongoing and iterative delivery of analytics.

Q&A

Next Lecture…

- Learn about:
  - Scrum
  - Kanban
  - Scrumban