Application Development in the era of Data Science and Artificial Intelligence

by Gavin Conran

Introduction

Although entrepreneurs have existed, in one form or another, in most cultures for many years, it was only towards the end of the Cold War that America's trust in this wealth generating paradigm was slowly adopted by European and other Western economies, thereby loosening their exising 'dirigisme' approach to managing their economies. At this time, the combination of entrepreneurism and technology, creating a critical mass, was embodied by two very different people, Bill Gates and Steve Jobs. Gates was technically gifted and an astute business man, but Jobs had an eye for design and in his mind the user experience was paramount, regardless of how advanced any underlying technology was. It was this focus that allowed the company he co-founded, Apple, to eventually create the iPhone, and with it a new era, not only in communications but also: commerce; politics; entertainment and fraud, to name just a few, and make Steve Jobs the most famous entrepreneur in history.

Building on the previous technology epochs of main frame computing (IBM), microchips (Intel), operating systems (Microsoft), personal computers (Dell), the Internet (Cisco), search engines (Google), E-commerce (Amazon), smart phones (Apple) and social media (Facebook) we now find ourselves in the era of Artificial Intelligence (AI). This paper explores strategies and tactics that will allow present day entrepreneurs, and managers in established organisations, to capatilise on the opportunities presented by AI, while, hopefully, avoiding many of the pitfalls.

We begin by capturing the problem of dealing with an uncertain world by outlining a well known example, the emergence of the Internet, and then explore how new approaches to strategy formulation and associated execution try to adapt to the unforseen changes that many businesses face when developing their offerings.

There follows an indepth discussion on the the techniques employed to allow for an adjustable approach to strategy execution, such as: the Lean Start-Up; formal Business Planning; Human Computer Interface (HCI) design; and data driven decision making all under the umbrella of an agile approach to product management.

The paper continues with a discussion on coding up applications for the era of Data Science and Artificial Intelligence. The main techniques explored will be Test Driven Development (TDD) for application development while concentrating on the scientific method and data to improve our mathematical models which lie at the heart of Artificial Intelligence applications.

The paper concludes with some thoughts on interfacing with supporting business functions in the customer facing, operational and administrative domains in order to scale operations and meet growing demand.

The Problem

Rarely do successful business ideas start ‘top down’ and most have ‘bottom up’ roots. The greatest top down failure and bottom-up success I witnessed, and described in my MBA thesis (1999) within the framework of Shumpeter’s notion of ‘creative destruction’, was the development and promotion of the Asynchronous Transfer Mode (ATM) standard for digital communications. In short, the Big Telcos would build ATM networks that would offer different qualities of service (QoS) in turn supporting voice, data and video communications with associated tiered pricing. Governments would set pricing limits and other regulations, the UN would set technology standards in the form of the I.T.U. and the established Telcos’ strangle hold on their respective telecommunications markets would continue into the new digital era. The Internet Protocol (IP) was a loose compilation of data communication standards, governed by the Internet Engineering Task Force (IETF), that emerged from academic and various government bodies to be the de-facto stantard of the Internet. It had two levels of Quality of Service, best effort, based on UDP, and guaranteed delivery founded on TCP, but there was no concept of ‘timing’ that was inherent in the higher quality levels of ATM which was deemed necessary to support voice and video communications. More importantly, there was no notion of pricing and, at the start, communicating over IP networks was free. As we now know and, with hindsight, unsurprisingly, IP won the day, as the Internet is the most widely used network in the world, and ATM is used only in niche networks, such as video networks found in the field of TV and film production.

Agile Execution for Emerging Strategies

It is for this and other reasons that Henry Mintzberg (from 1980) believed that successful business strategies ‘emerged’ rather than be prescriptively dictated from above; the traditional method of setting corporate strategy. At the time of my MBA the other main ideas and academics concerning strategy were: Michael Porter (from 1980) with his notions of ‘competitive strategy’ and ‘differentiation’ accompanied with a set of tools for practioners, including his ‘value chain’ and ‘five forces’ frameworks; Gary Hamel and C.K. Prahalad (from 1990) for their ideas on ‘core competences’ and ‘strategic fit’; Mintzberg with his notions of a ‘learning organisation’ which would be the source of organisations’ emerging strategy in a ‘knowledge economy’; and Clayton Christensen for his writings on ‘disruptive innovation’ (from 1995). Upon graduation, I remember, at the time, being filled with a zeal for creativity and innovation!

Since then a number of key thinkers have come to the fore, although Michael Porter is still viewed today as the world’s pre-eminent business thinker, including: W. Chen Kim and Renee Mauborgne with their ground breaking publication ‘Blue Ocean Strategy’ (2005) where they encourage entrepreneurs to combine differentiation and low cost, ignoring Porter’s warning of ‘being stuck in the middle’, to create new demand and make a new market; and Alexander Osterwalder and Yves Pigneur’s tool, the ‘Business Model Canvas’ (2008), coupled with Eric Ries’ insights on the process of innovation captured in ‘The Lean Start-Up’ (2011) to initiate new ventures.

Theory and practice suggests that the best approach to a new business venture, leaning on processes and ideas from the Lean Start-Up and the Business Model Canvas, is to quickly but iteratively design and build, however experimental, an application that solves a problem or meets a demand before investing time in a more formal and detailed business plan, which will be required if a venture is initially successful, wants to attract investment, scale operations and guide the business towards profitability.

Aligned with a top-down approach to strategy formulation was a top-down procedure for development, a process known as ‘waterfall’. The fundamental flaw of this Apollian, linear approach was the inability to adapt to change in a timely manner. By coupling ‘agile’ processes with a realisation that successful stategies emerge and evolve over time, oraganisations have the capability to adapt to a non-linear, business environment; an environment where Dioynsis thrives. Both the ‘Business Model Canvas’ and ‘The Lean Start-Up’ are approaches that enable the execution of an emerging strategy using agile techniques.

Data as a Core Competence

In the 90’s Hamel and Prahalad promoted two inter-connected ideas: the notion of a ‘core competence’ and ‘strategic fit’. The idea was to build a unique competence, ideally one that could legally be protected by patents, and then find markets where this competence added value, i.e. a strategic fit, giving a competitive edge. An example was optical fibre networks. The competence was in designing and building the optical components that allowed Gigabits of information to be passed down an optical fibre. These components where the bedrock of ‘core’, or trunk, networks. With the demand for ever higher bandwidth to the home, to support broadband services, similar optical technology could be used to roll-out fibre ‘access’ networks, which, traditionally had relied solely on copper networks to support a Plain Old Telephone Service (POTS). The problem was that it was difficult to stop cheaper, more nimble, competitors making similar market offerings. Investments in core competences were not returning a profit because the technologies were fast becoming commodities; in the language of Porter the barriers to entry were lowered.

Fast forward twenty years and the new, must have, competence is data. Of course, in able to compete an organisation will have to be skilled in data science, software engineering, maybe hardware engineering, design, marketing and selling on and off-line, but what makes their offering unique and competitive is the data in which their machine learning and artificial intelligence models are trained. In other words ‘data is the new gold’. This explains why organisations that generate vast amounts of user data, such as Facebook and Twitter, have huge market capitalisations. It also explains why the NHS’s patient data is worth so much as diagnostic applications, trained on vast amounts of patient data, would have a commanding position in any health market around the world.

The new world of data will be explored further in the section on ‘Data Science, Big Data and Data-Driven Decision Making’ as well as the section on ‘Building AI Applications’.

Agile Product Management

Agile is a set of software development principles created for effective and adaptive software development and offers the best chance of success by getting out of the building and engaging with customers and satisfying to their wants and needs. This is a key strength of agile as it seeks to be quick, resourceful, and adaptable.

The Agile Manifesto

Four core values and twelve principles of 'Agile' were orginally captured in the Manifesto of Agile Software Development in 2001 in the mountains of Utah by a group of like minded developers.

Four core values define the Agile philosophy for software development:

1. ***Individuals and interactions*** over processes and tools
2. ***Working software*** over comprehensive documentation
3. ***Customer collaboration*** over contract negotiation
4. ***Responding to change*** over following a plan

In addition, Agile has twelve principles:

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

Software Product Management in Practice

There are five main sections to Software Product mangement:

- Process
- Requirements
- Planning
- Monitoring
- Data Analytics and Real-Time Metrics (covered in a sepearte section)

Notes taken from Alberta Universitiy’s Specialisation on Software Product Management. The full course notes can be found in the folder ‘03_SoftwareProductManagement_CourseNotes’.

Process

A process organises the work of people into distinct high-level phases or stages to develop a software product, for example:

- Specification:
  Discover and define what the software is expected to do
- Design and Implementation:
  Construct the software
- Verification and Validation:
  Test for potential defects and review whether the product meets the client’s needs

Requirements

A set of specific descriptions to help capture a client’s needs, usally with a focus on the Human Computer Interface (HCI). Requirements can be mapped to features of the software product. When combined, processes and requirements are the backbone of any successful software project. There are techniques to properly elicit and express requirements. By spending some time to refine requirements, it is possible to detect potential errors in the product before it is even built. By clarifying ideas, development becomes focused and efficient.

Planning

Involves organising tasks and schedules to align software development activities with requirements, in order to make timely progress through the phases of the development process. Creating tasks and schedules requires identifying who should do the work, and estimating how long that work will take. As the goal is to adapt to change, planning occurs constantly and usually includes:

- Breaking down work into tasks
- Estimating time for tasks
- Assigning work

- Risk management
- Contingency plans

Monitoring

In order to ensure a project is on track it is important to actively monitor, analyze, and review a team’s progress. ***Velocity*** is a measure of a team’s productivity based on the units of work completed in a given time interval. Determining velocity allows more accurate time estimation of future work. If a project is under control it allows teams to **react to change** more easily and it gives a certain amount of ***transparency*** so eveyone on the team knows the status of the project.

Examples

An example of a project including Human Computer Interaction and Requirements Elicitation can be found in the folder ‘04_1_HCI_Requirements_Elicitation_Example’ with the outcome: https://myalliance.weebly.com/index.html

An example of a requirements document, including use cases, can be found in the folder ‘04_2_Requirements_Document_Example’. This document captures the requirements that is part of the business plan found in the folder ‘02_BusinessPlan_Example’.

The Lean Start-Up

A Harvard Business Review article by Steve Blank from 2013 titled “Why the Lean Start-Up Changes Everything” noted that in the past a business plan was written, which was pitched to investors, a team was assembled, a product was introduced to the market which was given the hard sell. Unfortunately for the majority of start-ups, somewhere along this timeline disaster would strike and the venture would fail.

As a solution, the article went on to introduce the notion of Eric Reis’ ‘The Lean Start-Up’ with, at its core, a philosophy of ‘fail fast’ and a commitment to ‘continual learning’.

The article stated the fallacy of the perfect business plan. Such a statement would have been greeted warmly by anyone who had written a business plan only to see the future deviate widely from any forecasts made in the document and the general feeling was succinctly captured by a quote from Mike Tyson “Everyone has a plan until they get punched in the mouth”.

To make a distinction between the approach taken by existing companies and start-ups the author claimed that one of the critical differences is that while existing companies execute a business model, start-ups look for a repeatable and scalable business model.

The lean method had three key principles:

1) Summarise hypotheses in a framework called a Business Model Canvas
2) Use Customer Development to test their hypotheses
3) Use Agile Development

Business Model Canvas

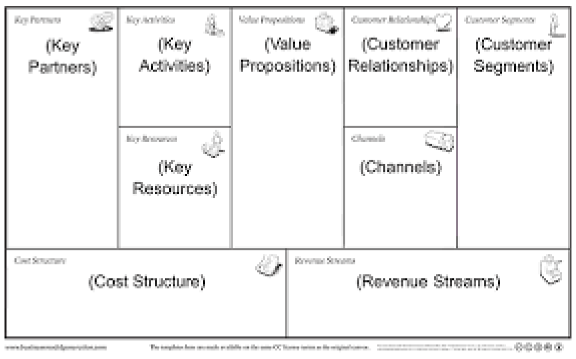

Illustration 1: Business Model Canvas

Rather than engaging in months of research and planning, entrepreneurs accept that all they have on day one is a series of untested hypotheses, i.e. educated guesses. So, instead of writing an intracate business plan, founders summarise their hypotheses in a framework called a Business Model Canvas, orginally devised by Osterwalder and Pigneur.

Customer Development

The main idea here is to leave the building and ask potential customers, users and partners for feedback on all elements of the business model, including product features pricing, distribution channels, and affordable customer acquisition strategies. Such interactions will allow the venture to maintain speed and nimbleness while rapidly assemble minimum viable products and elicit customer feedback. Once complete, the cycle would start again, testing redesigned offerings and making further small adjustments – iterartions - or more substantive ones – pivots – to ideas that are not working.

Agile Development

Agile development works hand in hand with with customer development. Agile development eliminates wasted time and resources by developing the product iteratively and incrementally - more on this in the Development section of the document.

Example

An example of a Lean Start-Up process, the course work of a 2012 MOOC, can be found in the folder ‘01_LeanStartUp_Example’.

Formal Business Plan

Once a product, solution or service has been built that is being paid for by at least one customer, although it is preferable to go to market with three reference customers, it is time to write a formal business plan and seek investment in order to scale operations and increase sales with the goal, first, to reach break-even and, secondly, proceed onto sustained profitability.

A well crafted business plan is no gurantee of success but it can act as a sanity check, just to make sure that the general plan has been thought through and all assumptions have been challenged, and it gives a target to aim for as the business grows.

Writing a Business Plan

A complete business plan usualy consists ot three components:

- Competitor Analysis
- Business Model
- Business Plan

Competitor Analysis

It can be fruitful to spend some time and energy researching and writing a thorough competitor analysis. It is analogous to creating a map of the current business environment. As well as including the positions of competitors that may reveal opportunities, threats, weaknesses and strengths (SWOT analysis), it should also include government regulations, laws and standards that a business operating in a market must adhere to. The Competitor Analysis can use 'Porter's Five Forces' as a framework to oraginse the data and the conclusions:

- Competitive Rivalry
- Threat of New Entry
- Threat of Substitution
- Buyer Power
- Supplier Power

It can become tedious to trawl through oceans of information and data but a diligently researched and compiled competitor analysis can be very revealing. The best tool for capturing the relevant data is a spreadsheet or a table in a word processing application, such as Microsoft Word.

Business Model

Time spent writing an interactive model will allow managers to run various scenarios where the demand, pricing and costs can be varied and the different outcomes captured graphically. The best tool for modelling is a spreadsheet.

Business Plan

This document will be read by investors so it is important the plan is written and presented in a way that they are used to. A typical business plan will have the following sections:

- Executive Summary
- What’s the big idea?
- Reference Customers
- Value Proposition
- Competitive & Market Forces (derived from Competitor Analysis)
- Go to Market Strategy
    - Sales
    - Marketing
    - Business Development
- Operations
    - Development
    - Production
    - DevOps
    - People
- Financials (scenarios derived from business modelling)
    - Profit and Loss
    - Cash Flow
    - Balance Sheet
- Final recap and Actions

Example

An example of a formal business plan with a competitor analysis and business model can be found in the folder ‘02_BusinessPlan_Example’.

Data Science, Big Data and Data-Driven Decision Making

Introduction

In this section, we explore the benefits of using ‘data’ within an orgaisation as well as make the distinction between data science, data-driven decision-making and big data.

As for the benefits, the utilisation of data resources can provide transparency and in-depth feedback to how a software product is being used by customers and data driven insights can become pivotal in deciding the future development of products and features.

Data Science

In 2013 Foster Provost and Tom Fawcett made a successful attempt to frame the emerging era of data within the context of business in their paper “Data Science and its relationship to Big Data and Data-Driven Decision Making”.

They defined Data Science as a set of fundamental principles that support and guide the principled extraction of information and knowledge from data. They noted that the broadest business applications were in marketing for tasks such as targeted marketing, online advertising, and recommendations for cross-selling. Data Science is also applied to general customer relationship management (CRM) to analyze customer behaviour in order to manage attrition and maximise expected customer value. The finance industry uses data science for credit scoring and trading and in operations via fraud detection and work-force management. Many firms, such as Amazon and Wal-mart, have differentiated themselves strategically with data science.

A data science perspective provides pracitioners with structure and principles but it also draws from many traditional fields of study including: causal analysis; statistics; and data visualisation.

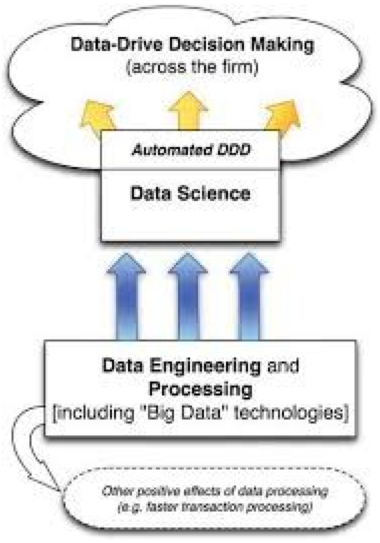
Data-Drive Decision Making
(across the firm)
Automated DDD
Data Science
Data Engineering and Processing
[including "Big Data" technologies]
Other positive effects of data processing
(e.g. faster transaction processing)

Table 1: Data Science in the context of closely related processes

Data Science and Data-Driven Decision Making

As discussed, data science involves principles, processes, and techniques for understanding phenomena via the, mostly automated, analysis of data. Data-Driven Decision Making – DDD – refers to the practice of basing decisions on the analysis of data rather than purely on intuition. The benefits of DDD to an organisation have been conclusively proven many times over.

Data Science and “Big Data”

Big Data can be referred to the data munging of data sets that are too large for traditional data-processing systems and therefore require new technologies, such as, Hadoop, Spark, Hbase and CouchDB. Big Data technologies are generally used in support of data science techniques, e.g. a number of large, disparate datasets consisting of structured and unstructured data can be mapped and reduced to create a data set of workable size. There is also little doubt that big data has proven beneficial to organisations.

Using the Insights

Firms in traditional industries are exploring new and existing data resources for competitive advantage by using data mining as a key strategic component. Facebook and Twitter have high valuations due primarily to data assets they are commited to capturing or creating. In the future, most if not all successful organisations will be using data driven insights to guide their product road-maps by feeding back their findings into the planning process for subsequent development iteartions.

Examples

Examples of Strategic Management analytics can be found at:
https://github.com/gavinconran/ESSEC/tree/master/StrategicBusinessAnalytics

Examples of Marketing analytics can be found at:
https://github.com/gavinconran/ESSEC/tree/master/MarketingAnalytics

Examples of Operations analytics can be found at:
https://github.com/gavinconran/Wharton/tree/master/OperationsAnalytics

Examples of Finance related analytics can be found at:
https://github.com/gavinconran/Wharton/tree/master/OperationsAnalytics

Building AI Applications

The interface between Product Management and Development is the Requirements Elicitation document, a specification of the system that customer facing roles can understand. The first step for the development team is to create an analysis document, which is a description of the same system but from a developer's point of view. Both the requirements and analysis documents go through a few rounds of amendments before both can be signed off. From the analysis comes a coarse grained system design, a pattern oriented architecture at the Java 'package' level. From the system design comes the fine-grained, detailed, object design usually captured in a class diagram. At this stage the developers can begin writing the code. Implementing Artificial Intelligence applications requires two main development procedures:

- Creating mathematical models and optimisation algorithms for the AI core
- Constructing software using Test Driven Development (TDD)

The Artificial Intelligence Core

Below is a list of artificial intelligent applications obtained from a post in Towards data Science by Abhishek Parbhakar in May, 2018:

- Large Scale Machine Learning (ML)
  developing systems that improve their performance with experience
- Deep Learning
  a subset of ML and a re-branding of neural networks
- Reinforcement Learning
  closed way of learning based on how human beings learn through rewards
- Robotics
  autonomous vehicles
- Computer Vision
  how a computer visually perceives its surroundings
- Natural Language Processing
  perception and understanding of human languages
- Recommender Systems
  what to read, what to buy, and whom to date
- Algorithmic Game Theory and Computational Mechanism Design
  interaction of multiple agents in the field of economics and social science
- Internet of Things (IoT)
  intelligent decision making from the gathering and processing of data from remote devices
- Neuromorphic Computing
  deep learning on a chip

The folder ‘05_ArtNet_DeepLearning_Example’ contains an example of an ML model, including a description of how to create a deep learning model using the TensorFlow framework.

Test Driven Development (TDD)

If artificial intelligence is to be used outside of the lab the model or algorithm will, probably, have to be encapsulated within an application so that it can be used directly by end users or embedded within a larger system, e.g. self-driving car. TDD relies on the repetition of a very short development cycle. Requirements are turned into very specific test cases, then the software is improved so that the tests pass. The following sequence is based on the book ‘Test-Driven Development by Example’:

1. ***Add a test***
   In TDD each new feature begins with writing a test. To write a test the new feature’s specification must be clearly understood. This is achived by having the requirements and exception conditions specified in a ***use case and user stories***. The test can then be written using a suitable ***testing framework***, such as Junit. This approach forces the developer to think about the requirements before writing the code rather than bolting on unit tests after the code has been written.
2. ***Run all tests and see if the new test fails***
   Validates that the ***test harness*** is working correctly, shows that the new test does not pass without requiring new code because the required behaviour already exists, and rules out that the new test is flawed & will always pass. The new test should fail for the expected reason.
3. ***Write the code***
   Write some code that causes the test to pass. The new code is probably far from perfect and will be further improved and honed in step 5.
4. ***Run tests***
   If all test cases pass, the programmer can be confident that the new code meets the test requirements and does not break or degrade any existing features. If the test fails the new code must be altered until the test passes.
5. ***Refactor code***
   The growing code base must be cleaned up regularly during TDD. The ***DRY (Don’t Repeat Yourself)*** principle dictates that duplication must be removed. As method bodies grow, they should be split and their parts carefully named so as to improve ***readability*** and ***maintainability***. Inheritance hierarchies may be rearranged to be more logical and helpful and perhaps to benefit from recognised ***design patterns***. By continually re-running the test cases throughout each refactoring phase, the developer can be confident that process is not altering any existing functionality.

Repeat
Starting with another new test the cycle is then repeated to push forward the functionality. ***Continuous integration*** helps by providing revertible checkpoints.

Examples of test and source code written using Test Driven Development (TDD) can be found at: https://github.com/gavinconran/Concurrency

Scaling and plugging into the rest of the organisation

Based on a simple rule of thumb - ***in the start up phase everyone works for development but in the growth phase everyone works for sales -*** the role of Product Management had been defined to be the interface with both sales and development; firstly, providing direction and requirements to development harvested and distilled from the market and secondly, guidance, sales materials & support for the customer facing and administration roles. As an organisation grows it will be helpful to return to the formal business plan and identify the roles which need to evolve and be added to the organisation. As Product Management and Development functions have been structured with scalability in mind growth should not be too taxing, but a growing organisation will now need more formal and experienced management and will probably need to have stand alone production, customer facing and admin roles:

Production / Operations

- Production Infrastructure
- DevOps

Customer Facing

- Sales and Support:
  - Direct or indirect sales channels
- Marketing:
  - Price, Place and Promotion
- Business Development:
  - Partnerships and Channel Management

Administration

- People Related:
  Hiring process and Skills development
- Finance Related:
  Profit & Loss, Cash Flow and Balance Sheet statements
  Financial management and setting budgets
- Legal:
  Patents, Copyrights, Customer & Employee Contracts and Partner Agreements

This approach and structure will give an organisation a fighting chance to scale and meet, hopefully, ever increasing demand