Architectural Thinking in the wild west of data science

Acknowledgements: Thanks so much to Kevin Turner for reviewing this document multiple times and for his valuable input.

Data Scientists tend to use ad hoc approaches. We see a lot of creative hacking of scripts in different programming languages on different machine learning frameworks distributed all over the place on servers and client machines. I'm not complaining about the way data scientists work. I've found myself in such highly creative modes many times when I really accomplished something significant.

Having a complete freedom in choice of programming languages, tools and frameworks improves creative thinking and evolvement. But at the end of the day, assets created by data scientists need to get into shape before finally delivered as there are major pitfalls as described below.

Technology Blindness

It is common sense to most Data Scientists that the actual technology doesn't matter too much from a functional perspective since models and algorithms used are anyway defined mathematically. So, the single source of truth is the mathematical definition of the algorithm. For non-functional requirements, this view doesn't quite hold. So, for example, the availability and cost of experts for a certain programming language and technology varies heavily. When it comes to maintenance, technology choice in fact has a major impact on project success.

Data Scientists usually tend to use the programming language and framework in which they are most skilled. This starts with open source technologies like R and R-Studio with its unmanageable universe of packages and libraries and its inelegant and hard to maintain syntax. The runner-up, on the other hand, is python with its well-structured and well organized syntax and associated frameworks Pandas and Scikit-Learn. On the other side of the tools spectrum are completely visual "low-code/no-code" open source tools like Node-RED, KNIME, Rapid Miner and Weka and commercial offerings like SPSS Modeller.

'The technology I know best' is fine for PoC (Proof of Concept), Hackathon or start-up style of projects – but when it comes to industry and enterprise project scale, some architectural guidance on technology usage must be in place – howsoever it may manifest.

Lack of Reproducibility and Reusability

Given the problem statement above it might be obvious to you that uncontrolled growth of data science assets in an enterprise setting can't be completely tolerated. In large enterprises, a lot of churn happens in terms of projects and human resources; e.g. external consultants with specific skills are only hired for a short period of time attached to a project. Usually, if a human resource leaves the project, also their knowledge is gone. Therefore, it must be ensured that Data Science assets aren't just a collection of scripts implemented in

different programming languages laying around in a variety of different locations and environments. Because of the non-collaborative nature under which many data science assets are developed, it follows that reusability of those assets is often limited. Ad-hoc documentation, poor code quality, complex and mixed technologies and a broader lack of expertise are the main drivers for this problem. Once these issues are addressed, assets become reusable and dramatically improve in value. E.g. if uncoordinated, every data scientist re-creates the ETL (Extract – Transform – Load), Data Quality Assessment and Feature Engineering pipeline for the same data source leading to significant overhead and poor quality.

Lack of Collaboration

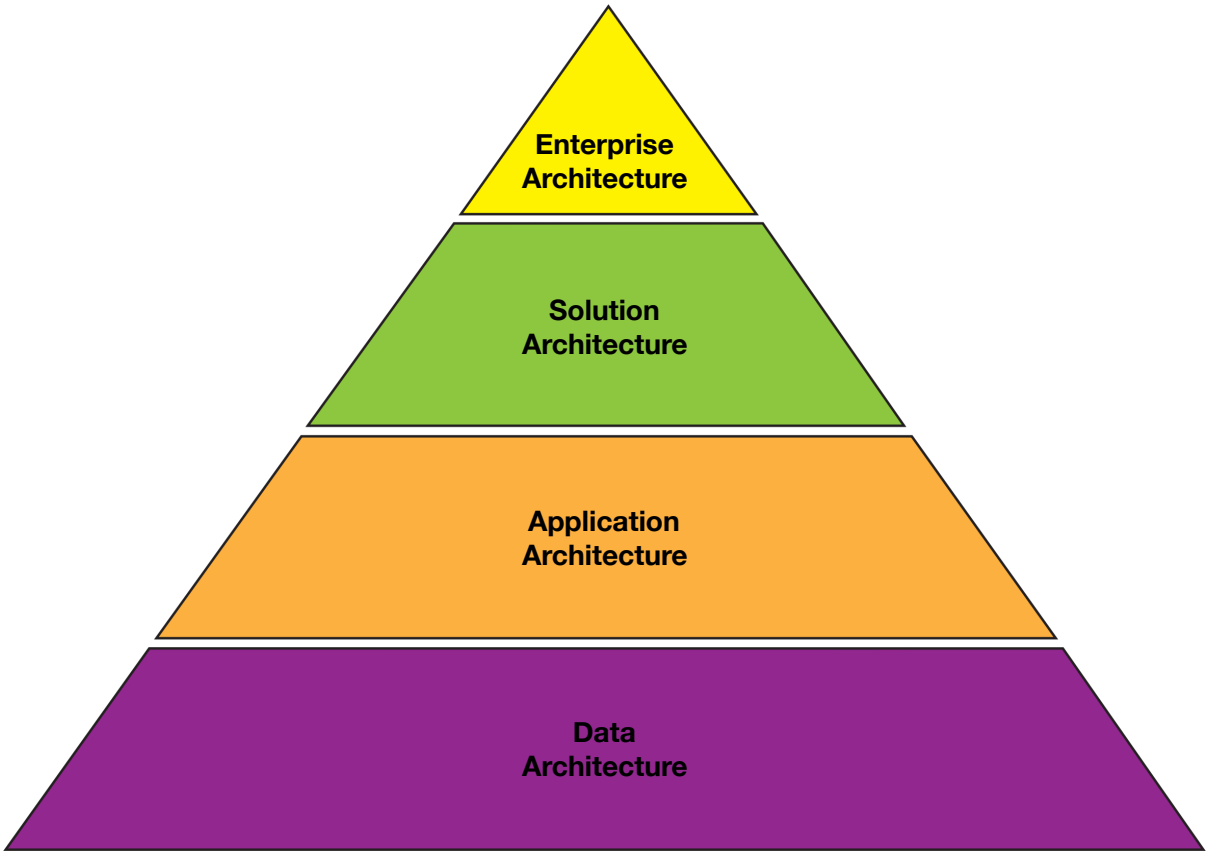
Data Scientists are great thinkers. Common sense tells them that brains do not scale. Therefore, Data Scientists tend to work alone at their own pace in their own manner. If they are stuck, social media like "stackexchange.com" or alike is one of their best sources to get help – maybe it is ignorance or maybe only lack of equally skilled peers - the technically best data scientists often don't excel in collaboration. For outsiders, it might look like they are following the mind-set "Après moi, le deluge", assets created aren't shared and organized in a reusable manner. Poor documentation – if present at all – and scattered components make it hard to retrace and replicate pervious work. Therefore, a common asset repository and minimum guidelines for proper documentations are indispensable.

Suboptimal architectural decisions

Data Scientists are often 'hackers' with linear algebra skills and some understanding of business. They are usually not trained software engineers and/or architects. As stated before, data scientists tend to use the programming language and frameworks in which they are most skilled and progress rapidly to solution without necessarily having non-functional requirements (NFRs) like scalability, maintainability and human resource availability in mind. Therefore, I emphasise the need of a Solution Architect or Lead Data Scientist role attached to every major data science project, to ensure that NFRs are properly addressed. To support such a role with a pre-defined architectural and process framework is of great help. But first, let's have a look how a traditional enterprise architecture fits into data science projects.

How much architecture and process is good for a data science project?

Before we answer this question let's start with a short review on traditional enterprise architecture and later evaluate how an architectural methodology and process model fit in.



Architecture hierarchy. Souce: IBM Corporation

At the top of the pyramid stand's the enterprise architect. He defines standards and guidelines valid across the enterprise. Some examples include:

- Usage of Open Source software is only allowed if having permissive licenses
- REST calls always need to use HTTPS
- Usage of NoSQL databases requires special approval from the enterprise architecture board

The Solution Architect works within the framework the Enterprise Architect defines. He defines what technological components fit the project or use case. Example of such a definition are:

- Historical data must be stored in a DB2 RDBMS
- For high-throughput structured real-time data Apache Spark Streaming must be used
- For low-latency, real-time video stream processing IBM Steams must be used

The Application Architect then defines his application within the framework of the Solution Architecture. Examples of such a definition are:

- The UI is implemented using the MVC pattern
- For standard entities, an object relational mapper is used
- For complex queries, prepared SQL statements are used

Finally, the data architect defines data related components like:

- During ETL data needs to be de-normalized into a star-model
- During ETL, all categorical and ordinal fields must be indexed

So where does our all-mighty, creative data scientist cowboy fit in here? First, we try to define with of the roles above a Data Scientist may partially take or with which roles he might interact.

So, let's again have a look at the roles from top to down. To get more illustrative, let's take a metaphor from urban design. An Enterprise Architect is the one who designs a whole city. He defines sewerage systems and roads for example. A Solution Architect then would be the one designing individual houses whereas an Application Architect designs the kitchen and a Data Architect the electrical installation and water supply system.

Finally, let the Data Scientist be responsible for creating the most advanced kitchen ever! So, he won't just take an off-the shelf kitchen. He will take individual, ready-made components but also create parts from scratch where necessary. The Data Scientist will interact with the Application Architect mostly. If the kitchen has special requirements the Data Architect might be necessary to provide the infrastructure.

So, keeping this metaphor in mind for a second. How do you think the kitchen would look like if created by the Data Scientist alone? It will be a very functional kitchen with a lot of features. But most likely lacking some usability. So, for example, to start the oven you need to login to a Raspberry Pi and execute a shell script. And since all parts have been taken from different vendors including some custom-made hardware the design of the kitchen is ugly. Finally, there is a lot of functionality but some of it is not needed and most of it is undocumented.

Going back to IT – this example illustrates the answer to the original question. So where does our all-mighty, creative data scientist cowboy fit in here?

The Data Scientist would rarely interact with the Enterprise Architect. He might interact with the Solution Architect but will work closely together with the Application Architect and Data Architect. He doesn't need to take over their roles but he must be able to step into their shoes and understand their thinking. As Data Science is an emerging and innovative field, the Data Scientist must speak at eye level with the Architects (which is not the case for an Application Developer or Database Administrator) to transform and influence the Enterprise Architecture.

Let's conclude with an example to illustrate what I mean by this. Consider architectural guidelines that R-Studio Server is the standard Data Science platform in the enterprise and that all Data Science projects must use R. This software was approved by the Enterprise Architect and the on-premise R-Studio Server self-service portal was designed by the Solution Architect. Now the Data Scientist finds a Keras code snippet in python using the

TensorFlow backend which awesomely pushes model performance to the moon. This code is open-source and maintained by one of the most intelligent brains in AI. He just needed an hour or so to plug this snippet in to his data processing pipeline running on his laptop (yes, he prototypes on his laptop since he really doesn't like the crappy R-Studio Server installation provided to him). So, what do you think should happen here?

In the old days of the all-mighty architects in an enterprise the Data Scientist would have been forced to port the code to R (using a less sophisticated Deep Learning framework).

But here's the potential. In case the Data Scientist wants to use this code snippet, he should be able to do so. But if this is done without guidance, we end up in the wild west of data science.

So, let's have a look at existing process models and reference architectures to see if and how we can merge the traditional field of architecture with the emerging field of data science.

An overview on existing process models for Data Science

CRISP-DM

CRISP-DM, which stands for Cross-industry Standard Process for Data Mining, is the most widely used open standard process model – if a process model is used at all, of course. CRISP-DM defines a set of phases which make up a data science project. Most importantly, transitions between those phases are bi-directional and the whole process is iterative, this means once you've reached the final stage you just start over the whole process and refine your work. The illustration below emphasizes on that nature:

The CRISP-DM process model. By Kenneth Jensen, based on: ftp://public.dhe.ibm.com/software/analytics/spss/documentation/modeler/18.0/en/ModelerCRISPDM.pdf - Creative Commons Attribution-Share Alike 3.0 Unported license

In my opinion this process model is already a good start. But since it is a process model only, it assumes that the architectural decisions on the technology used and the NFA's are already addressed. This makes CRISP-DM a very good model in technologically settled environments like traditional enterprise data warehousing or business intelligence projects.

In a rapidly evolving field like data science it is not flexible enough.

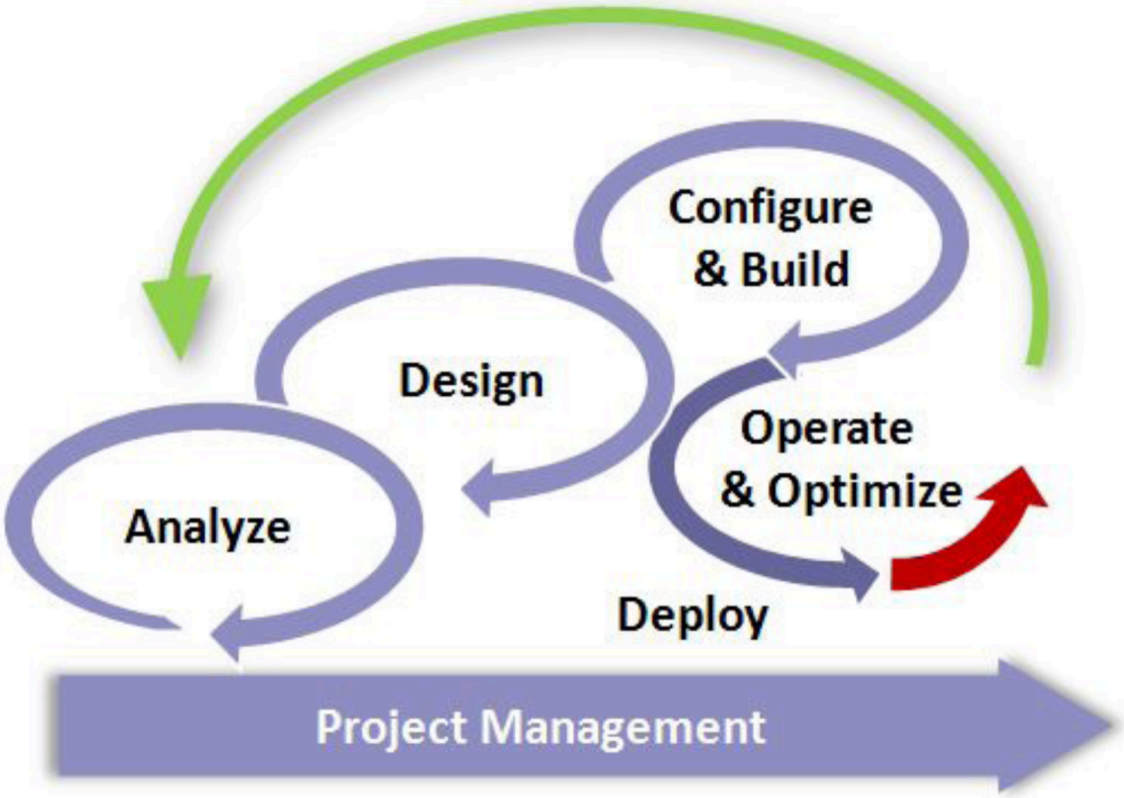
ASUM-DM

Due to shortcomings in CRISP-DM, 2015 IBM released the Analytics Solutions Unified Method for Data Mining/Predictive Analytics (also known as ASUM-DM) process model. It is based on CRISP-DM but extends it with tasks and activities on infrastructure, operations,

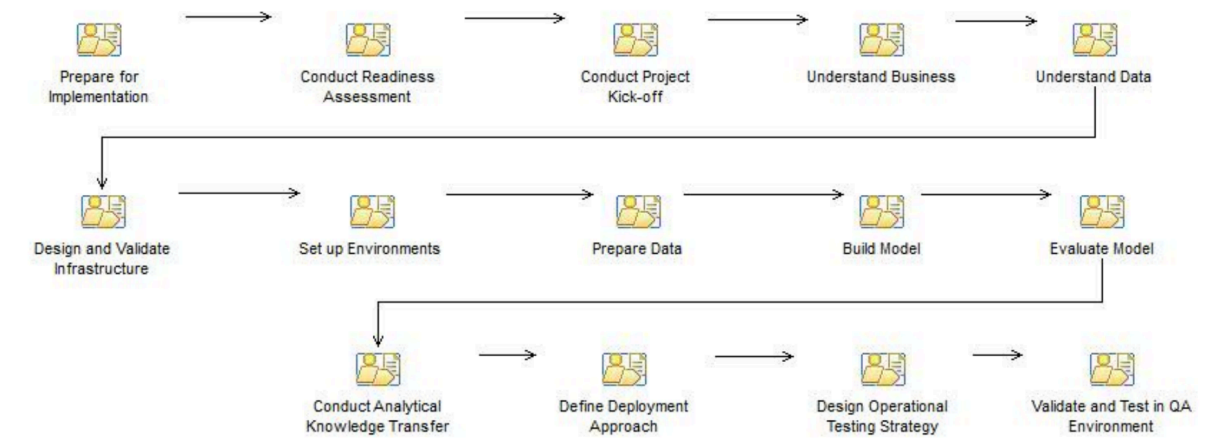
project, deployment and adds templates and guidelines to all the tasks. There is an open version of ASUM-DM available here https://www14.software.ibm.com/webapp/iwm/web/pick.do?source=swerpba-basimext&lang=en_US, the full scale version is only available to IBM clients – you can inquire here: asmarket@us.ibm.com to become one :-)

By the way, ASUM-DM is part of a more generic framework called Analytics Solutions Unified Method (also known as ASUM) providing product/solution-specific implementation roadmaps covering all IBM Analytics products.

ASUM-DM borrows the process model from ASUM which is illustrated below:



Analytics Solutions Unified Method (ASUM) Process Model. Source: IBM Corporation



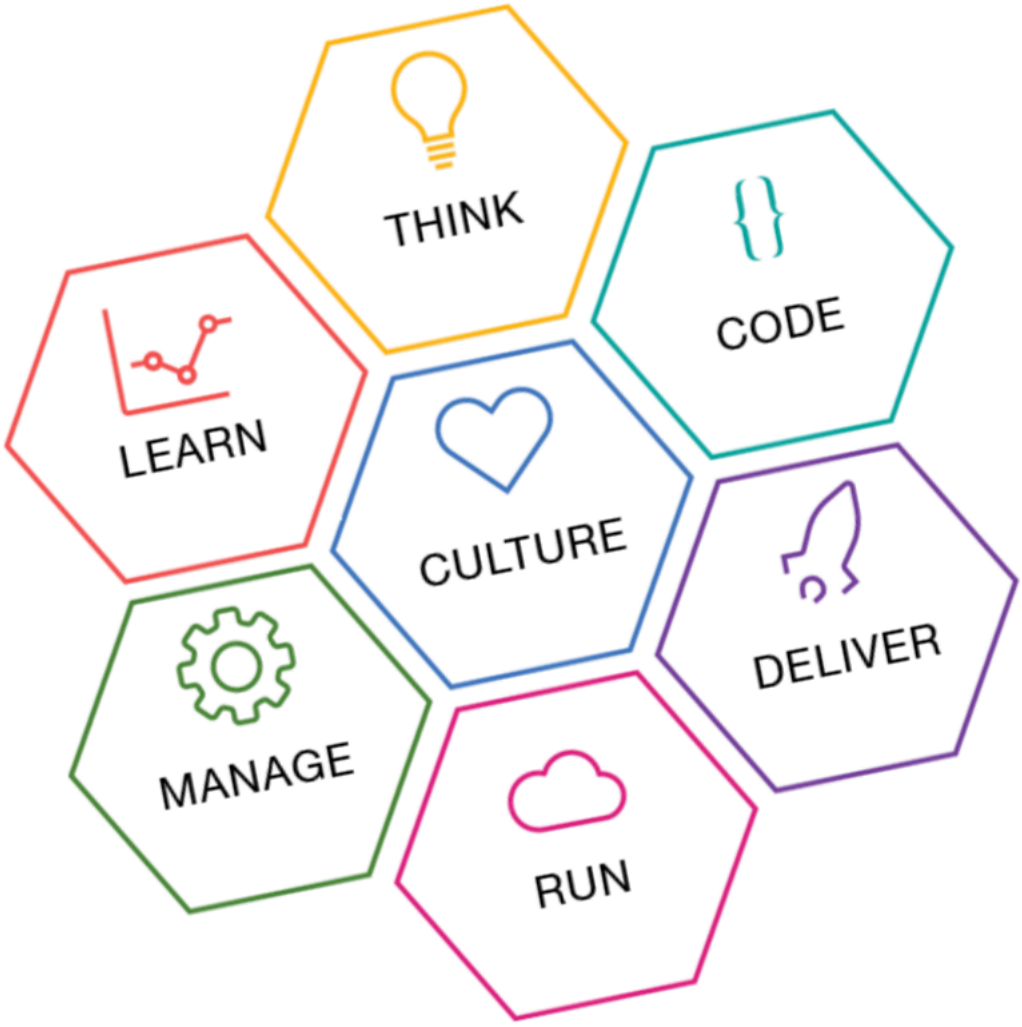
Analytics Solutions Unified Method (ASUM) Process Model Detail. Source: IBM Corporation

IBM Cloud Garage Method

When the Manifesto for Agile Software Development http://agilemanifesto.org/ was published in 2001, heavy processes like Waterfall or V-Model went out of vogue. The main reason for this paradigm shift was the software development crisis in the 90s where software development just couldn't keep up anymore with rapidly growing expectations of business stakeholders on time-to-market and flexibility.

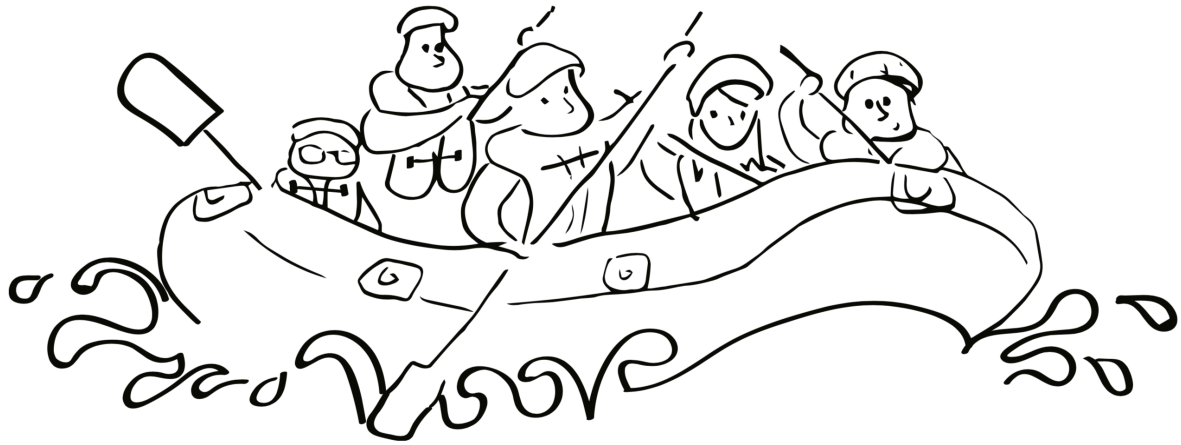
Because enterprise clients often have a hard time transitioning to agile processes, IBM created the IBM Cloud Garage Method. An agile software architecture method tailored to enterprise transformation.

Again, this method is organized in different stages as illustrated below:



The IBM Cloud Garage Method. Source: IBM Corporation

Key thing to notice here is that cultural change is in the middle of this hexagon. This means without cultural change the method is doomed to fail. This is important to keep in mind. In the context of Data Science, we have a head start since Data Scientists tend to favour lightweight process models if used at all.



In the IBM Cloud Garage Method, every practitioner sit's in the same boat. Source: IBM Corporation

Here's a summary of all the six phases surrounding the cultural change:

Think

Design Thinking is the new requirement engineering. Design thinking has its roots already in the 60s but IBM was one of the major contributors to apply this method to the IT industry.

Although usually stated in more complex terms, Design Thinking in my opinion has only one purpose:

Switch your brain into creative mode. Therefore, writing and drawing is used over speaking and typing. By stepping back you'll be able to see, understand and create the bigger picture. Design Thinking has the user experience in mind and a clear emphasis on the business behind the offering. So these key questions are answered:

Who – For whom do we build the offering?
What problem are we trying to solve?
How are we going to solve the problem?

The outcome of every think phase is the definition of a minimum viable product (MVP).

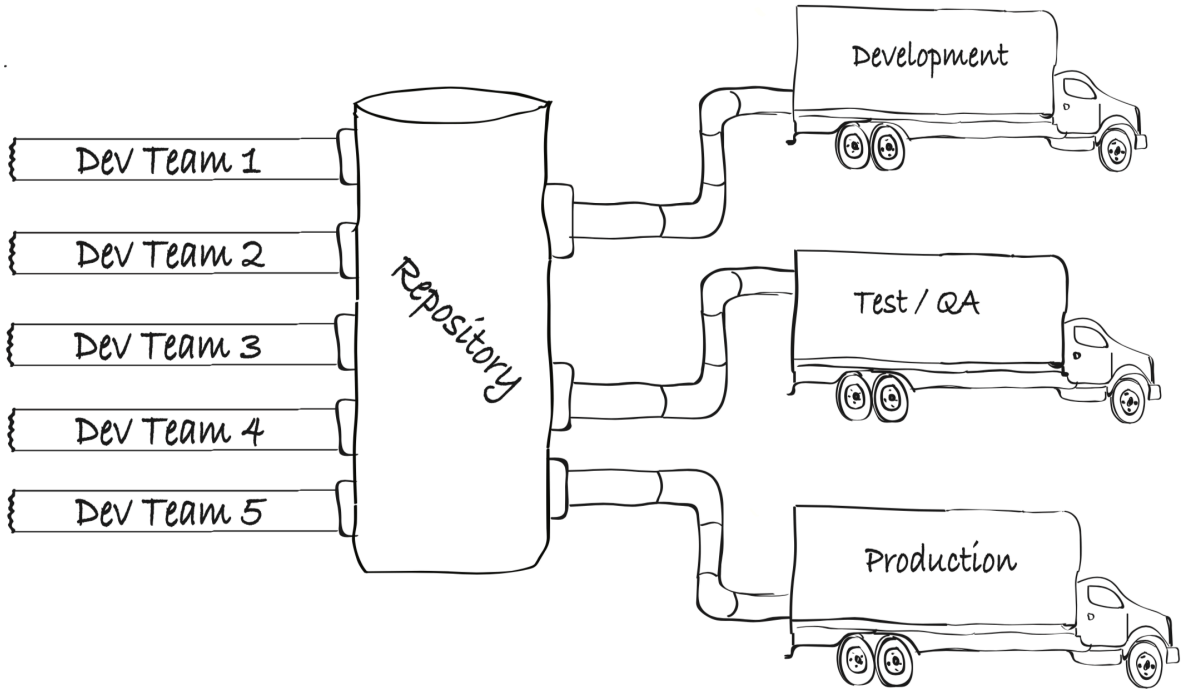
Code

The platform cloud revolution is the key enabler for fast prototyping. You can get your prototype running in hours instead of days or weeks. This allows you to shorten the iteration cycle by an order of magnitude. This way user feedback can be gathered daily. Best practices for this phase include:

- daily stand-up meetings
- pair-programming and test-driven development
- continuous integration
- automated testing
- refactoring to micro services

Deliver

Prerequisites for daily delivery are two things. First, build and deployment must be fully automated using a tool chain. Second, every commit to the source code repository must result in a fully, production ready product which can be tested by end users at any time. Cloud based solutions are tackling this requirement and let developers concentrate on coding.

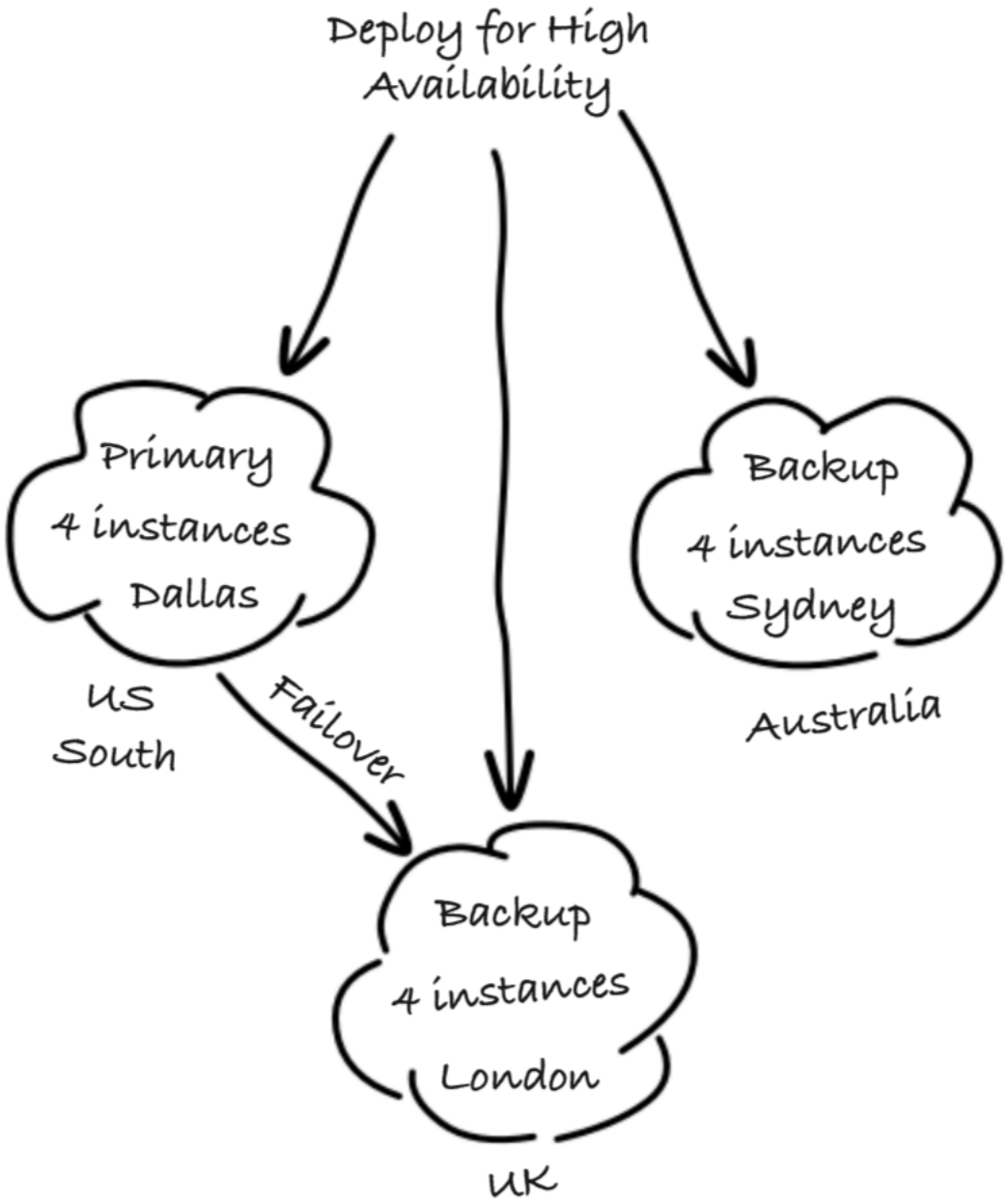


Continuous Integration and Continuous Delivery. Source: IBM Corporation

Run

Once using a cloud runtime, operational aspects of a project are handled by cloud services. Depending on requirements this can happen in public, private or hybrid clouds and at an infrastructure, platform or service level. This way often the operations team can be made obsolete and developers can concentrate on adding value to the project. Best practices for this phase include:

- readiness for high availability
- dark launches and feature toggles
- auto-scaling



High-Availability, Auto-Scaling and Fault-Tolerance in an intercontinental cloud deployment. Source: IBM Corporation

Manage

Since premising on fully managed cloud runtimes, even adding intercontinental high-availability / failover, continuous monitoring and dynamic scaling isn't a challenge anymore and can be simply activated.

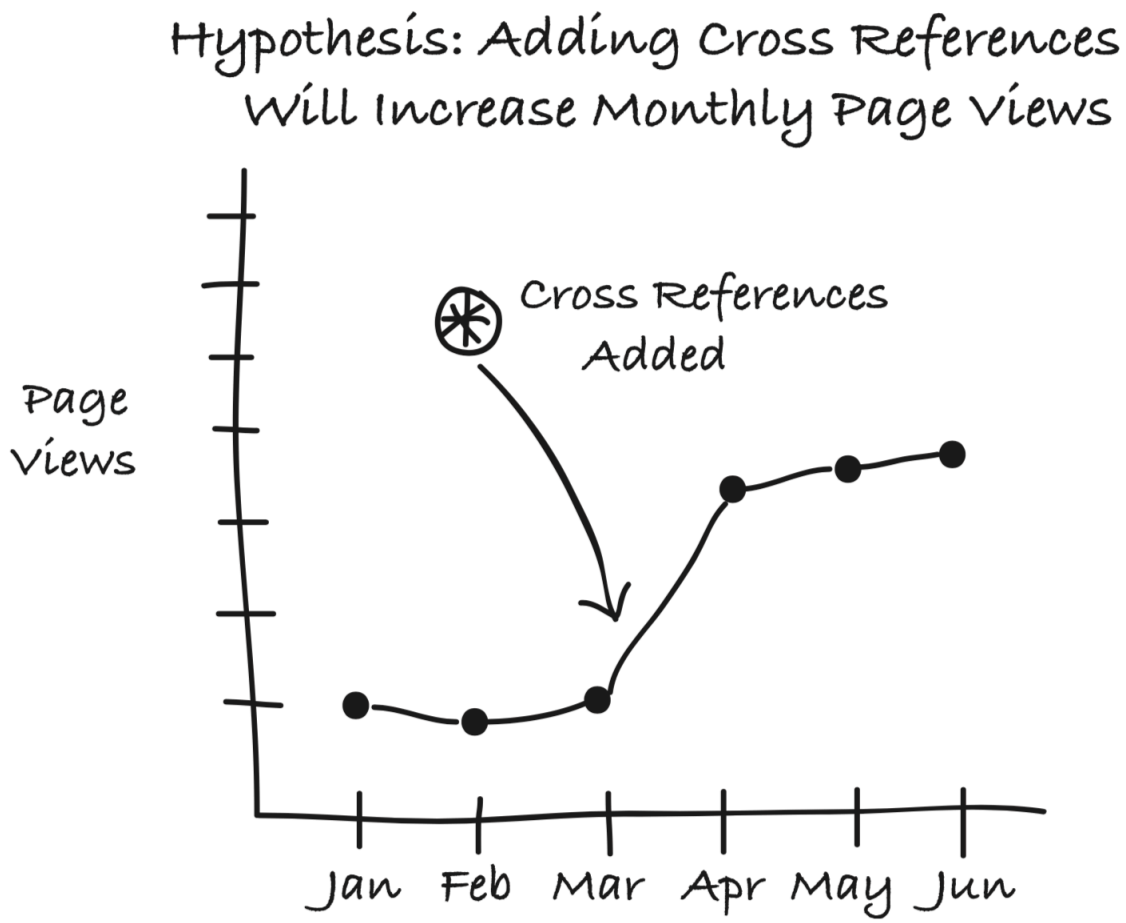
Best practices for this phase include:
- automated monitoring
- fast, automated recovery

- resiliency

Learn

Due to the very short iteration cycles and continuous user feedback, hypotheses can be tested immediately to generate informed decisions and drive findings which can be added to the backlog for further pivoting. Best practices for this phase include:

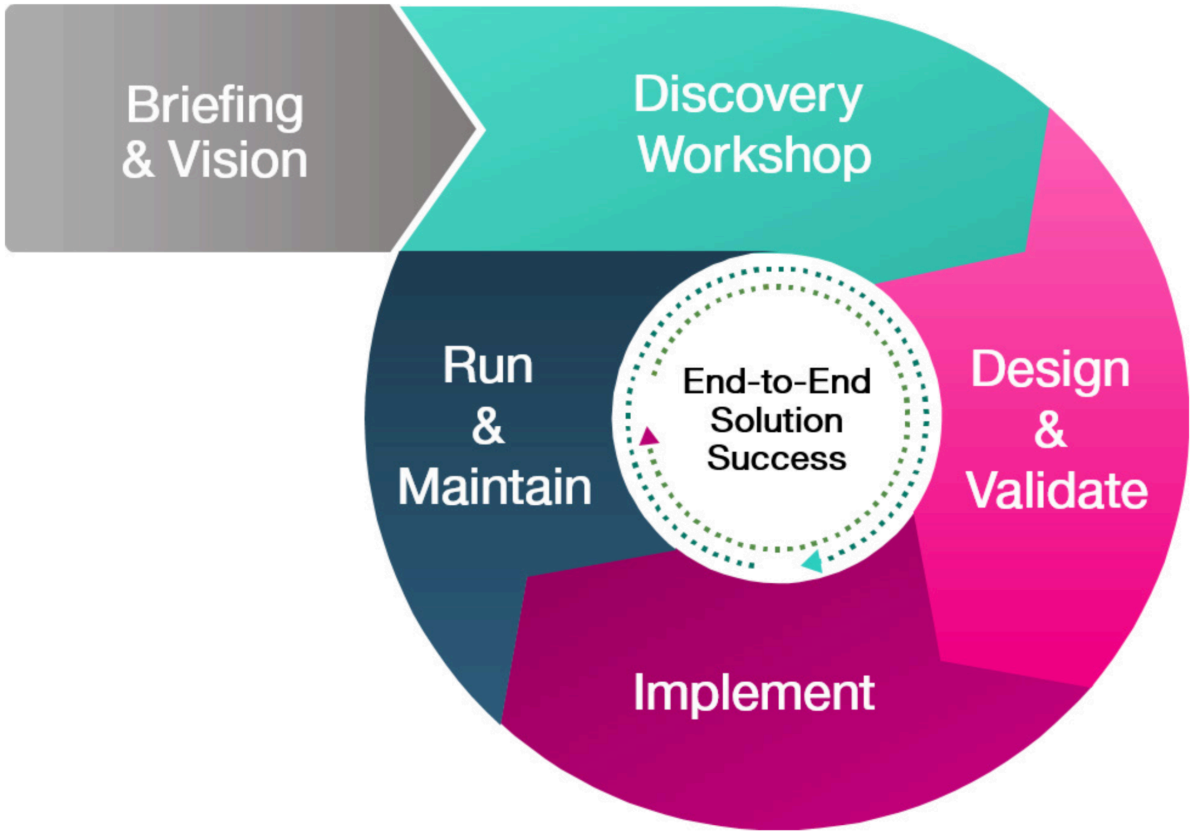
- A/B testing
- hypothesis driven development
- real-time user behaviour analytics



Evidence based hypothesis testing example. Source: IBM Corporation

IBM DataFirst Method

Although usually bound to an IBM client engagement included in the DataFirst Method Design Engagement Offerings – the IBM DataFirst Method is an instance of the IBM Cloud Garage Method – specifically targeting IT transformation to get infrastructure, processes and employees ready for AI. For more information visit ibm.biz/DataFirstMethod



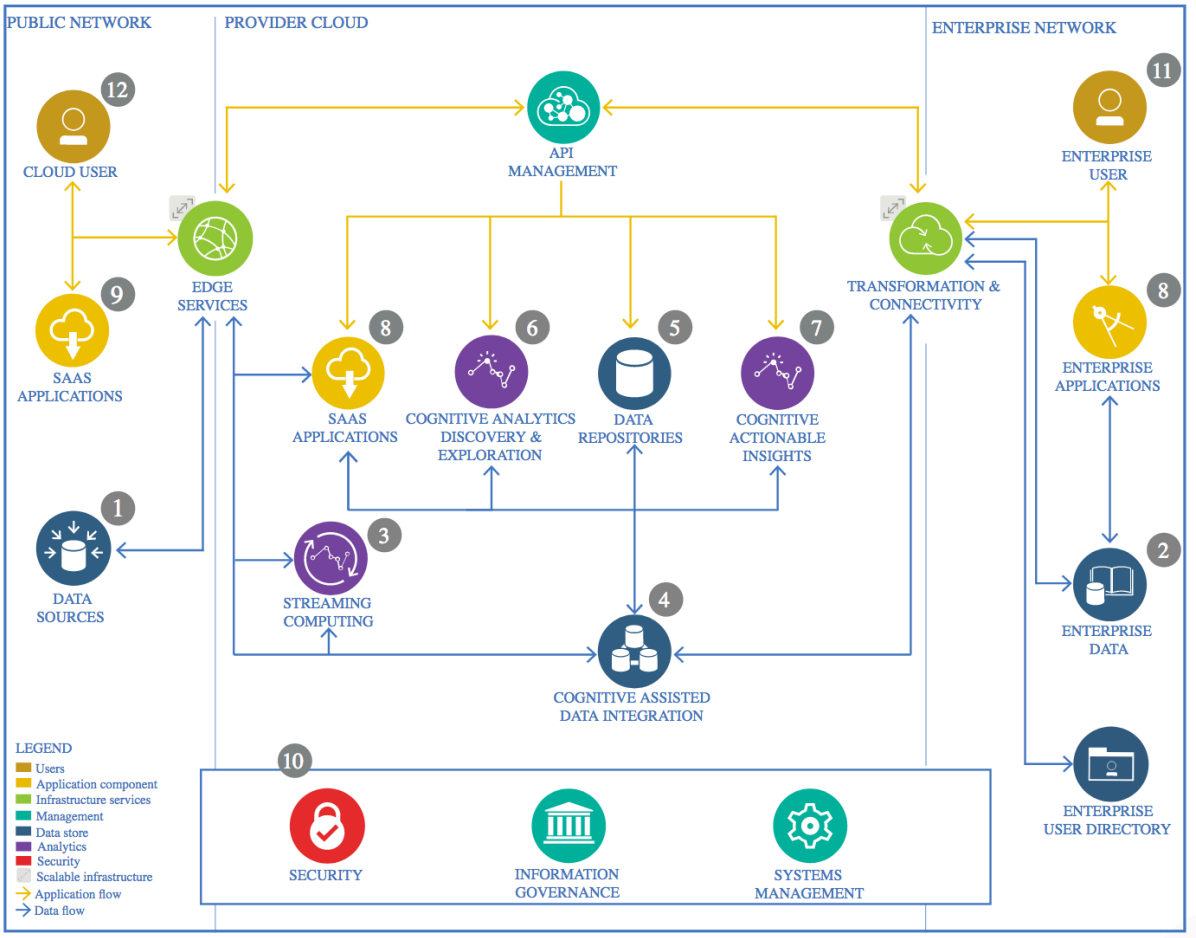
IBM DataFirst Method Process Model. Source: IBM Corporation

The IBM Data and Analytics Reference Architecture

Every project is different. Every Use Case needs different technical components. But they all can be described in abstract terms. The following list enumerates and explains those:

1. Data Source - in internal or external data source which includes Relational Databases, Web Pages, CSV, JSON or Text Files, Video and Audio data
2. Enterprise Data – cloud based solutions tend to extend the enterprise data model. Therefore, it might be necessary to continuously transfer subsets of enterprise data to the cloud
3. Streaming analytics – current state of the art is batch processing – since decades. But sometimes the value of a data product can be increased tremendously by adding real time analytics capabilities – since most of world's data loses value within seconds. Think of stock market data or the fact that a vehicle camera captures a pedestrian crossing a street
4. Data Integration – here your data is cleansed, transformed and if possible, downstream features are added
5. Data Repository – this is the persistent storage for your data
6. Discovery and Exploration – here you get an idea what data you have and how it looks like
7. Actionable Insights – This is where most of your work fits in. Here you create and evaluate your machine learning and deep learning models

8. Applications / Data Products – models are fine but their value rises when they can be consumed by the ordinary business user – therefore one needs to create a data product
9. Data Products doesn't necessarily need to stay on the cloud – they can be pushed to mobile or enterprise applications
10. Security, Information Governance and Systems Management - This important step is forgotten easily – it's important to control who has access to which information for many compliance regulations
11. An enterprise user is part of the architecture since his requirements may differ from a public user
12. Cloud user's requirements may differ from those of enterprise users



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

Conclusion

So now since you have an overview over all current state of the art methods and process models for data science on the cloud it's time to concentrate on a method most useful for individual data scientists in the field who want to improve their methodologies, want to minimize architectural overhead and positively influence enterprise architecture bottom-up. I call this the Lightweight IBM Cloud Garage Method of DataScience. I'll explain this method in my next article. So stay tuned!