INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




WARP Business Intelligence System

VIVIANA RAQUEL SILVA BATISTA
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

WARP Business Intelligence System

Implementing continuous integration in a Business Intelligence project

Viviana Raquel Silva Batista

Dissertation to obtain the Master of Science degree in
Computer Science, Specialization in
"Sistemas de Informação e Conhecimento"

Supervisor: Ana Maria Neves Almeida Baptista Figueiredo

Jury:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, February 2017

To my mother and father for all of the motivation,
to my anchor in the hardest of times, Hugo,
this would never be possible without you.

“Never forget what you are, for surely the world will not.
Make it your strength.
Then it can never be your weakness.
Armour yourself in it, and it will never be used to hurt you.”

George R. R. Martin

Abstract

Continuous delivery (CD) facilitates the software releasing process. Because the use of continuous integration and deployment pipelines, allows software to be tested several times before going into production. In Business Intelligence (BI), software releases tend to be manual and deprived of pipelines, versions control might also be deficient because of the project nature, which involves data and it's impossible to version. How to apply CD concepts to BI to an existing project where legacy code is extended and there is no version control over project objects? Only few organizations have an automated release process for their BI projects. Because due to projects nature it is difficult to implement CD to the full extent. Thus, the problem was tackled in stages, first the implementation of version control, that works for the organization, then the establishment of the necessary environments to proceed with the pipelines and finally the creation of a test pipeline for one of the BI projects, proving the success of this approach. To evaluate the success of this solution the main beneficiaries (stakeholders and engineers) were asked to answer some questionnaires regarding their experience with the data warehouse before and after the use of CD. Because each release is tested before going into production, the use of CD will improve software quality in the long run as well as it allows software to be released more frequently.

Key words: *Business Intelligence, Continuous Delivery, Continuous Integration, pipelines, Data Warehouse, releases*

Resumo

Continuous Delivery (CD) permite que as *releases* de software aconteçam em qualquer momento sem problemas associados, utilizando *pipelines* de integração e de *deployment*. Desta forma, o *software* é testado várias vezes antes de ser instalado em produção. Em *Business Intelligence* (BI), as *releases* são tendencialmente manuais, sem *pipelines* e devido à natureza do projecto (dados) o controlo de versões tende a ser inexistente. Como aplicar o conceito de CD num contexto de BI a projetos de grandes dimensões, com *legacy code* extenso e sem controlo de versões? Apenas algumas organizações têm um processo automático de releases para os seus projectos de BI, porque devido à natureza dos projetos que envolvem dados, é dificil implementar CD. Tendo em conta os estes factores, o problema foi abordado por etapas, em primeiro lugar procedeu-se à implementação de um controlo de versões, que se adapte às necessidades da organização. O passo seguinte foi a criação do ambiente necessário para prosseguir com a instalação de *pipelines* e para terminar, a terceira etapa, consistiu na criação de uma *pipeline* de teste para um dos projectos de BI, comprovando assim o sucesso da solução proposta. Para avaliar o sucesso desta solução os principais beneficiários (*stakeholders* e engenheiros) foram convidados a preencher questionários, que permitem avaliar a sua experiência com o *data warehouse* antes e depois da utilização da solução proposta neste trabalho. Como cada *release* é testada antes de ser instalada em produção, garantindo que possíveis erros ja fora encontrados previamente, o uso de CD melhorará a quailidade do *software* a longo prazo e permitirá que as releases ocorram com mais frequência.

Palavras-Chave: *Business Intelligence, Continuous Delivery, Continuous Integration, pipelines, Data Warehouse, releases*

Acknowledgements

To my boyfriend, Hugo, for always helping me to overcome all obstacles and for never letting me give up on the longest nights that do not seem to have an end. To my parents for their help and motivation during these three years of hard work and keeping my spirits up, for their comprehension on missed family dinners, my mad mood and above all for loving me through the tougher of times.

To my friends who showed me that academic life is nothing without their company, who accompanied me every weekend in our second home. To Ana Almeida (May our reunions always happen over ice cream), Bruno Gomes, Miguel Vilaça, Nuno Lima e Ricardo Costa, for all the hours of laughter provided during the masters, for always keeping my spirit up and not letting me forget that I was able to finish this stage.

To Farfetch for giving me the opportunity to belong to an ever growing and innovative family. To my BI team for helping me in this project and for giving me the opportunity to learn more each day.

To Professor Ana Almeida, for the support during the writing of this document and for helping me in all the necessary moments. To all the teachers that lead me to through this path and to success.

To the very noble institution that welcomed me for seven years, Instituto Superior de Engenharia do Porto, for letting me grow up and become the adult that I am.

Table of Contents

List of Figures

List of Tables

Acronyms and Symbols

List of Acronyms

BI	Business Intelligence
CI	Continuous Integration
CD	Continuous Delivery
ETL	Extract, Transform and Load

1 Introduction and Contextualization

This chapter purpose to contextualize and introduce the problem, focusing on business concepts, setting up the problem in the organization, also identifying the target audience, made assumptions and possible implications. Finally, problem definition, its main objective and the expected results are presented.

1.1 Context and problem

1.1.1 Business Concepts

The concepts considered for this project are the following:

Business intelligence (BI) – According to Hans Luhn, BI is a "*System that provides means for selective dissemination to each of its action points in accordance with their current requirements or desires.*" (Luhn 1958). This paper was first published in 1958.

Data warehouse (DW) – Following the definition of Ralph Kimball, "*A data warehouse is a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making*." (Kimball & Caserta 2015)

ETL – Extract, Transform and Load are the processes the data goes through between the operational source systems and the DW/BI (Kimball n.d.).

Agile – Considering the Agile Manifesto description, Agile is a methodology that focus on delivering good products to customers by operating in an environment that values the people, embraces modeling and planning, understanding the workflow changes through its lifecycle (Beck et al. 2001). More than that, it's a mindset and a behaviors set used to leverage agile practices and techniques. (Noone n.d.)

Continuous integration (CI) – CI is “a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily—leading to multiple integrations per day” (Duvall et al. 2007).

Continuous delivery (CD) – The concept of continuous delivery is defined as a discipline where software is built in a way that it can be released to production at any time (Fowler 2013).

In figure 1 it is represented the Continuous Integration and Continuous Delivery relation.

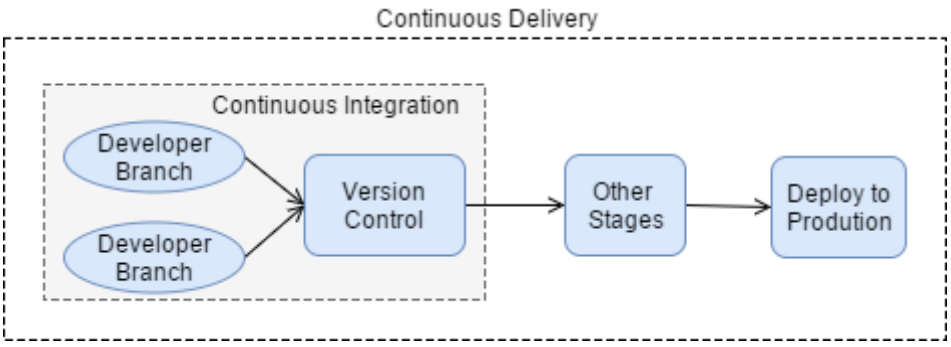


Figure 1 – Continuous Integration and Continuous Delivery relation

WARP - Expression used in the organization to represent a fully automated release process. It can be represented by layers, starting from the center with test automation, followed by CI, CD and the last layer is WARP, as shown on Figure 2. Each superior layer depends on the success of the previous, the innermost layer represents the implementation of automatic tests, if successfully completed, the team proceeds to the next layer, CI. Here the team focuses on using CI correctly together with tests automation. The next step in archiving WARP is CD, releases, in this stage occur through pipeline but its ownership is of DevOps instead of the team. At the final layer the team has complete ownership of its release pipeline, therefore they are responsible for releases frequency and any possible error found during this process.

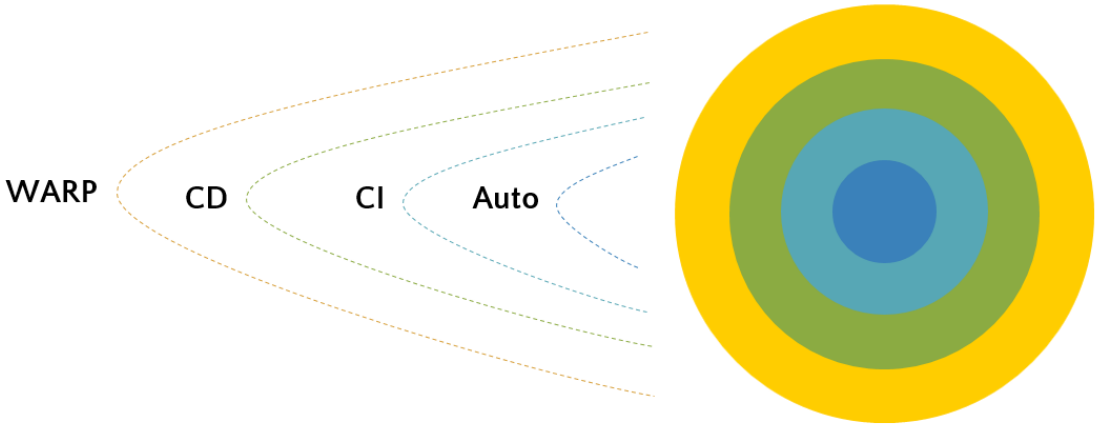


Figure 2 - WARP Layers

1.1.2 Engineering purpose

Purpose of this solution

The company begun the activity as a startup in 2007. Since then, it had an exponential growth, leading to the lack of organized documentation and consequently information is dispersed, being many times centered on individuals, in their implicit knowledge. This leads to an extra difficulty in understanding and manipulating projects.

Currently, the company uses CI in every project, CD is used partially though company projects. Starting to use CD in big projects with a large amount of legacy code and outdated architectures can be complex and time consuming. Consequently, using CD tends to be overlooked over value deliverance. Regarding the BI team, CD wasn't even previously considered because of the small size of the team and the product. Now that the team as grown from five to twenty people, it's unreliable to have a manual release process and no version control over the code developed, due to errors caused by releases and delay in delivering value.

The purpose of this solution is to provide an autonomous automatic release process for a BI project, developing a CD solution that includes a solution for project CI and the deploy process automation. With this solution, it's possible to bridge the lack/delay in releases which this project currently faces, as well as reduce the errors induced by the current release process. The solution will be implemented to the data warehouse and ETL process, the cube will be addressed in a near future.

Mission requirements

One of the problems software organizations face, is how quickly it's possible to deliver features to users. Currently, organizations focus on requirements management and the impact on the delivery speed, being fast delivery of new features the main objective. In most of these cases, different software development methodologies and other software lifecycle stages are forgotten, as in the release process. With this, the need for an agile methodology solution urges, one that enables developers, testers, build and operations personnel to work together effectively. (Humble et al. 2010)

The customer

This project has, as customers, the BI teams and consequently their stakeholders, which are the main beneficiaries since software will be delivered faster and less prone to errors later on. It is expected that the resulting process is used by the teams during development stages and later while releasing it. In addition, stakeholders are the ones that will benefit the most from the process, since their requests will not only be addressed faster but also be released quicker.

Satisfying customer requirements

Through this solution, the customers will use a CD approach for their development cycle, with this, version control and releases stop being a set-back on a task lifecycle, they will be just another step in the cycle. This solution will lead to faster and more frequent deliveries of value without leading to errors on the ETL process, since there is minimal human interventions during the process and better code coverage quality, as version control will keep track of software code and allows comparison between files, identifying differences, and merging the changes if needed prior to committing any code.

Definition of value for the customer

The ability to easily release software with control and less chance of error, with the addition of having a system easy to install on any machine. Also, the availability of a version control system that allows him to keep up with every alteration made on the software. Both features will be of great value for the customers.

Adaptation of existing projects

Having the knowledge of previous developed technologies with many years of use in the global market, it's wiser to use these technologies instead of creating/using new ones. Still, the goal is to have a solution design especially for this project, so using an existing solution, even though it's possible, isn't an option. Due to different necessities, all through the project in different stages, it's essential to use different approaches in each stage, there isn't a solution that fits all the requirements.

1.1.3 Point of View

The proposed solution should be able to support the follow requirements present in a CD solution:

- CI, using a version control tool over every system object. This will provide an historic of every modification of an object during its existence and allow multiple developers to develop several tasks at the same time while merging every change during the development;

- Business-facing tests and technology-facing tests and regarding the development process. These must always run successfully before releasing the feature into production, which allows the customers to predict how the new release will affect the system;

- Automated deploy/release for wanted features, each release should be relative to one task, following the CD methodologies.

Personal views of stakeholders

As referred before, the BI teams are the main customers of this solution, releases are quicker and more frequent. Business stakeholders are the ones that ask for new features and having them frequently released with success makes them an interested part in this solution.

1.1.4 Assumptions

Currently, the organization has a team focused on CD that is responsible for managing team's pipelines. The software used for deploys and orchestrating jobs are the same, Octopus and Jenkins, respectively. But other tools can be used to help each team fulfilling their needs, being the use of the previous tools are mandatory. An additional conditioning factor is the availability of developers and release expert for this project that due to commitment with the organization can't be 100% focused on the project.

1.1.5 Engineering information

For this project the supporting information used consists on the values presented in the book "*Continuous Delivery Reliable Software Releases Through Build, Test and Deployment Automation*"(Humble et al. 2010) and also the guidelines expressed in the Agile Manifesto (Beck et al. 2001).

1.1.6 Implications

The pipeline developed needs to be constantly updated and improved to accomplish the team's needs. Also, the goal for this pipeline is to be used in the development cycle of each team so, monetarization of successful releases during its usage is important to find new ways to improve and realizing the importance of them.

1.2 Problem

The purpose of the present chapter is to expose and explain the problem, for which the need to carry out this project arose. BI value chain and delivery doesn't fall under the traditional waterfall development umbrella. (Larson & Chang 2016) The need for real-time analyses has increased over the years (Halper 2015), so it's necessary to deliver value faster. Projects that deliver solutions quickly to business users and incrementally improve within each release have greater value delivered continuously. (Kenney n.d.) Deployment of BI system are complex and require a formal support to ensure the health of the system, without it, the risk of failure increases. Incremental releases allow a controlled deployment and recognition of value by the stakeholders (Larson & Chang 2016).

A BI project includes different objects coming from reports and dashboards, databases, ETL process and any other object that can be important to the project. Ownership of this object's is responsibility of the BI team, therefore any changes and deployments are managed by them. The amount and diversity of objects brings complexity to each object deployment since there isn't a "fit all" solution.

Reports and dashboards don't have the same properties as databases or the ETL process. Consequently, this group needs to be managed independently from the other two, regarding both version control and deployment. On the other hand, databases and ETL have dependencies among them. The latter accesses to two type of databases, the operational and the data warehouse. The first one only provides data to the process, almost no writing is required, while the second is the target of the writing process and the reading process by the reports and dashboards. Any release related with databases can have impact on the ETL and the same goes for ETL changes, which are reflected on the database. Another important concern is about data, which is the core attribute of a BI project. Its integrity and security is complex, so it is necessary to take precautions regarding these issues and the company's data management guidelines. Because there are projects with different architectural objects the development of an automatic deployment process is complex and costly.

Before CD was mainstream and used by big organizations, the term CI was used. It was popularized in the '90s and since then is broadly used for software development (Ståhl & Bosch 2014). CD is the natural evolution of CI, the term was first used on 2010, by Jez Humble and David Farley. Per them, CD is "*a pattern for getting software from development to release*" automatically and ready to install anywhere necessary (Humble et al. 2010). This pattern is a deployment pipeline, which is, an automated implementation of an application build, test, deploy and release process. Every change made on an application triggers a new instance of the pipeline. Which will create the installers, tests them proving they can be released. Each stage of the pipeline concluded with success proves the stability of the installers and proves to be a release candidate. Once all stages are successful the installers can be release (Humble et al. 2010).

CD is considered an agile methodology, it's even a part of the agile manifesto, and "*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software*." (Beck et al. n.d.) CD practices lead to lower risks in releases, accelerated value delivery, improved quality of products/services, lower costs and motivated, happier teams (Humble n.d.). Some studies have proven the benefits described and showed above. This benefits include is also the gain of independence, any developer can deploy an application to test or production environment (Itkonen et al. 2016).

Applying the concepts of CI to data warehouse development has historically been more complex (DANIEL PERIANEZ 2014). Automation tools target conventional development, therefore, are not designed to address the requirements of BI (Swoyer 2016). However, BI projects can harvest the benefits of CD, in a thoughtful manner. Because BI is a data-driven

process, it's needed to find tools that are tailored to deal with the team's needs, having in mind the agile and CD principles.

Unlike other components of the project, data can't be easily integrated. Either the developer creates an insert script or data is copied from a production-live environment. This means that it isn't possible to apply versioning to data. These scripts, should be held with attention since their behaviour alters the data, and so, possible rollbacks can be hard to do. Regarding other objects such as XML packages, reports and DML, files version control and CD is easier to maintain, but concerning XML packages it isn't possible to merge two files since each one's components have distinct identifiers. In this case the solution is to replace the previous version with the new one, this requires the team to have an organized when editing this files and methodical with their commits and merges.

1.3 Objective

The main purpose of this work is to create a CD release pipeline, providing BI development teams with a version control over project objects, automatic testing of features and automatic deploy. Tests should range from feature tests to regression tests and user acceptance though several controlled environments allowing tests to eliminate errors in earlier stages of release, which results in frequent releases with little time spent on them and less prone to errors.

This solution is aimed at BI teams and consequently stakeholders, since more frequent releases means more features and requirements developed, which faces the biggest setback with the current release process. Stakeholders often see their requests unfulfilled timely because releases are delayed due to any difficulty. The absence of version control and automatic deploys for database objects as well as ETL process, affect the ease of integration, delivery of features and in some cases the release can mislead the ETL process, thus delaying data availability.

1.4 Expected / achieved results

With this project it is expected to obtain an automated release process that can be executed several times quickly and successful with minimal human interaction, providing safer and agile releases, unlike the current release method which is manual, meaning releases aren't frequent, therefore value isn't being delivered.

Releases should occur several times a day, each one representing a new feature deployed to live, therefore this process should be quick and feedback, after each release, frequent, in order to understand the impacts the release have in the system. Thus releases are expected to run within minutes, feedback should happen frequently and during the pipeline stages.

To measure the success of this project, stakeholders will evaluate their experience with the product after project finalization. It's expected that their requests are attended quickly so their product evaluation will improve over time. Pipeline duration and number of releases will also be measured to test project success.

2 Value analysis

In this chapter project value will be analyzed in three parts, starting by concept development and proposition, followed by the CANVAS and finally the way a project value should be evaluated according to Verna Allee and Porter Michael.

2.1 Value offer

2.1.1 New concept development model

NCD acts as a common language, providing a vocabulary for understanding the activities that occur at the front end, enabling companies to discuss front end with more understand and coordination between all (Methods et al. 1996). The goal is to clearly identify requirements, business plans and definition of market for a new project. NCD divides the front end into three distinct areas the engine, the inner area (wheel) and the rim. The engine provides power to the innovation process, which consists of the company values, leadership and culture, drives the inner part. The model has a circular shape to indicate that ideas flow, circulate and iterate among the five elements of the inner part. This part consists of five activity elements (Opportunity identification, Opportunity Analysis, Idea Generation & Enrichment, Idea Selection, and Concept Definition) and has two starting points for projects. They either begin by opportunity identification or idea generation & enrichment or leave the front end of the process by entering new product development process or technology state gate. The third element consists of the external environmental factors that influence the engine and activity elements, these consist of outside world influences, competitors influence, customers and organizational capabilities.

Opportunity identification

On the opportunities that might be pursued, these are usually driven by business goal and is focused in areas where the company might want to participate in. An opportunity refers to a business or technology gap, released by designer accident that exists between the current situation and the envisioned future. The company, or individual, might not yet have a solution for this gap.

There are a few methods that can be used to ease the assessing of opportunities such as:

- Road mapping, is a technique for planning technological capabilities. Because it's in graphical form, supports strategic alignment and communication (Dictionary 2017). Its key value is the mapping process that provides a platform for sharing wisdom between the team's resources, capabilities and skills. It's easily understandable to people who aren't part of the project team (Methods et al. 1996).

- Technology and customer trend analysis, is the research of trends among customers and technology to find relevant and competitive opportunities for the project.

- Competitive intelligence analysis, is essential to developing a business strategy, that produces actionable finding, since it collects and analyses information on competitive trends outside one's company (Methods et al. 1996).

- Market research, is the research of the surrounding market, on which the project falls.

- Scenario planning, is an approach preparing possible future scenarios and find decision that would otherwise be ignored (Methods et al. 1996).

Manual releases are a reality for the BI teams at the organization, it’s prone to error leading to less frequent releases. This gap allowed to the identification of an opportunity, device a solution of CD that fits the needs of the teams, improving frequency of delivery and of value. To represent the opportunity correctly the best techniques are the creation of a roadmap, technology trend analysis and scenario planning. The first will allow to map and clarify, among stakeholders, the requirements, priorities and timing of development. Since it's a technological project, it's important to analyze trends in development techniques, understanding their benefits and where they are lack comparing to other. Also, if any of these new technologies serve any purpose for the company. Finally, design possible scenarios will bring to light questions that would only arise during testing stages, e.g. if two team members develop over the same object how will the solution handle it?

Opportunity Analysis

To confirm if the opportunity is worth pursuing it's necessary to analyze the opportunity, formally or iteratively, making technology and market assessments. Because this isn't an in depth analyses, uncertainty about technology and market will remain, only business capability

and competency are assessed. This element happens more than once, teams will loop back to this element for new features identified in the concept definition stage.

Some techniques used for this are the same as the previous element, but it's possible to identify four main ones:

- Strategic framing, finding where the opportunity fits within the company's market and technology strengths, gaps, and threats (Methods et al. 1996).

- Market segment assessment, evaluating detailed information of the market segment, to show why it represents a great opportunity. It's key to determine market size, growth rates and market share of competitors. This is impacted by economic, cultural, demographic, technological and regulatory factors. Therefore, companies rather evaluate big opportunities.

- Competitor analysis, finding and defining who the main competitors are, by evaluating their capabilities, and new products needed to achieve competitive advantage.

- Customer assessment, determining major customer needs and if they are met by current products (Methods et al. 1996).

To analyze the present opportunity, the best technique to use is the strategic framing. Seeing the opportunity aim is to deliver value internally for the organization, it's important to understand how delivering value affects them business. Currently, stakeholders request new features but due to a slow and difficult delivery process, these are developed but released later when the feature is not as relevant. Strategic framing, will allow to identify how the opportunity fills the gap of the company and how it will improve its feature delivery to stakeholders. Achieving this, turns the opportunity in a strength instead of a thread, since stakeholders tend to ask for the feature to other teams trying to find a workaround.

Idea Generation & Enrichment

Just like opportunity identification, this element is an entry point in the new concept development model. Its objective is birth, develop and maturate ideas. These go through change all through the process and have many iterations as changes rise. To enrich the idea, contact with costumers and users are common. The process of idea generation can be formal but may also emerge from non-formal processes.

The techniques adopted for this element, by teams or individuals, range from more formal ones to more creative. The following are just some of the ones used widely.

Theory of Inventive Problem Solving (TRIZ), which enhances creativity by getting individuals to think beyond their experience and use solution from other areas of science (Methods et al. 1996).

- Methods for identifying customer needs, such as ethnographic approaches and lead user methodology. The first, a qualitative approach, to do a qualitative research with emphasis in an entire culture. Lead user methodology is a research method built around the concept that the richest understanding of new products needs is held by just a few Lead Users (Herstatt & Von Hippel 1992).

- Analyzing the market and business needs/issues, and also, new technology solutions that might fit the company's future plans and provide technological advances. This is possible by increasing technology flow through internal and external linkages and/or partnering.

- Companies that promotes and encourages employees to spend time investing in new ideas, or either by award them to stimulate the generation of new ones, tend to have more success in idea generation since they come from the ones working directly with the product. Besides this, the company can provide an idea-bank available with easy access to product or service improvements.

- Having a formal role for coordinate ideas generation and handles them through the business units. With this type of management, it's important to keep track of successful ideas and their implementation, therefore having measurable goals becomes crucial. This will offer inspiration to other employees and prove non-believers that new ideas can be a major advantage for the company.

- Rotating jobs frequently helps knowledgement sharing and extensive networking, it creates mechanisms for communication of core competencies and capabilities.

Idea generation is widely promoted among the organization where the project is to be developed. Which opportunity emerged from the lack of continuously successful releases, in the BI teams. Because it's a project with large number of possibilities, having frequent brainstorming meetings is crucial for the success of project, making sure to get as many ideas as possible for which software to use or possible architectures.

Idea Selection

Idea selection is an everyday life activity, when selecting an idea, the company must pursue the biggest business value, critical to future health and success. There is no correct and perfect solution for finding the best idea, in this model, ideas have limited information so understanding of its possible future is very little. Even though there isn't a solution for all selection type, some techniques are generally used.

- Individual judgment is one of the most commons since its natural and its part of the initial selection process. But it's ineffective, without some formal decision process most ideas disappear (Methods et al. 1996).

- Formal idea selection begins with someone looking at an idea whose information is limited, more information should be pursued if the idea is considered attractive. For this, it is important to communicate with the originator of the idea, this way he feels informed otherwise he might feel his idea was being taken away. Communication between the decision maker and the originator is important to understand the struggles between both, the originator might want to follow-up with the idea but he might feel overwhelmed with work and not able to invest further in it. Therefore, ideas picked up from collection process should always have a feedback and frequent reviews, this will stimulate creativity and more ideas. After gathering information, the idea is discussed with the people involved to clearly understand who own and who rely on the processes being discussed. In this stage, the decision maker needs to adopt a positive attitude and ask what idea will move the company forward, or what can be changed so it can become an advantage, instead of discarding less attractive ideas. Once again encouraging creativity.

- Portfolio methodologies based on multiple factors using numeric indicators, not just financial. For cases where the idea is only mildly explained, traditional techniques, like this one, have proven to be unsuitable since an idea is dismissed, by the company, if not considered profitable. But other techniques, not focused in traditional ones, like strategic fit and leverage can help in the decision-making process.

- Use of options theory to evaluate projects analyses market risk, which consider the probability distribution of the cash flow stream or its independent revenue and cost components actually enhances the option's value (Methods et al. 1996). The power of this techniques can only be real when critical thinking is involved around the assumptions of what is the idea value.

Like said previously, because of all the possible scenarios for the project, a high number of ideas exist. Selection of ideas occur base on a formal process, after choosing a possible idea more information about it is gathered and in some cases experimental tests to prove its success and adaptability to the solution.

Concept Definition

The final element and only exit of the new process develop model is the concept definition. In this element, the innovator makes a compelling case for investment in the opportunity chosen, also known as "win statement" or "gate document" (if the exit is done by the technology stage gate (TSG)). Investors can only decide based on qualitative and quantitative information given by the innovator. Because it's a critical decision to make, proportional to the financial investment needed, some companies define guidelines to follow when choosing to approve the opportunity. The criteria depend on the nature, type and what risk are the decision makers willing to make for the opportunity. If the case is not accepted, the concept returns to NCD or becomes dormant. Once it's on the NCD the opportunity is review and modified to become stronger, when it becomes dormant it can be a challenge to keep it alive and relative.

To move a project from NCP to New Product Development or into Technology stage gate it's critical to develop a formal project proposal and a business plan if possible.

The process of concept definition can be done using, among others, the following techniques.

- Goal deliberation approaches, in this approach a necessary number of measurable objectives, regarding business goals or outcomes expected. Not meeting this objectives can result in project termination (Methods et al. 1996). It's up to the project owner to clear any misinformation and assumptions, this should be made during the deliberation as a disagreement management tool.

- Setting criteria for what describes what an attractive project looks like. These will be used to explain the concept to everyone and might be found a good tool in idea generation and idea selection.

- Rapid evaluation of high-potential innovations, this is possible with the use of short cycles of requesting funding, testing specific technical details, review by an experienced innovation team, finding potential business value, the nature of the innovation, a plan to quickly test the highest-risk element of the concept and the identification of potential sponsors for the project are evaluated (Methods et al. 1996). This is allowing for a rapid assessment of concepts as well as their possible value for the company.

- Rigorous use of the TSG for high-risk projects, this can be accomplished out of the NCD process, for some cases this is essential, if the technology activities were mostly structured and with few risks, or if there was a business decision to specifically pursue a particular technology (Methods et al. 1996).

- Understanding and determining the performance capability limit of the technology [2].

- Early involvement of the customer in real product tests, this type of technique provides rapid feedback and more detailed needs from users.

- Partner outside of areas of core competence (Methods et al. 1996).

- Focus (Methods et al. 1996).

- Pursue alternative scientific approaches (Methods et al. 1996).

For this solution, the best approach is to define acceptance criteria, making it possible to have a list of requirements that needs to be fulfilled. The criteria is defined based on the customer needs and technology requirements.

2.1.2 Value, value for the customer and perceived value

Value

According to Oxford Dictionary, value stands for the importance or worth of something (Dictionaries 2017). It might represent a monetary worth, but value is more than that. At present type value has a different value for different theoretical contexts (NICOLA et al. 2012).

For companies, creating value is key for success, business activity is about exchange of value between the customer and the company, being an exchange of services (intangible) or having the customer pay the price for a product (tangible). Value isn't static but rather dynamic and should change with the needs of the customer. This can only be achieved if the company has its value well defined and explicit for the public and knows how to deliver this value for the intended customers. This concept is hard to conceive and conceptualize, to get to the value the companies must right the right questions: what does they do, who is their market segment, and how can they describe their value for someone with no background. For the client, this mean the value of what they are acquiring is explicit and they are aware of it. On the other side, a badly defined value can make the customer feel disappointed and ending up rating the company poorly.

Value for Customer

When a customer acquires a product, there is a personal perception placed on it, meaning the value of the individual consumer can be different from the value defined by the company. This happens when the customer associates an advance to a certain product again others on the market, consumers tend to buy products with the highest value for customer. According to Woodall (Woodall 2003), *"captures a range of associated, existing concepts, all of which use similar names and imply a similar idea"*. For customers, this attribute presents itself as a benefit, companies try to have the best value for customer on the market, but on the other side, if companies badly market their product, customers start to create expectation that fails to be fulfilled.

Perceived Value

In the Cambridge Dictionary, perceived value stands for the value of a product/service based on the needs of the customers, rather than its monetary value (Woodall 2003). This means, the value of the product, on the customer perception, is not associated with its price but rather the fulfilment of expectations created previously to acquis ion,

All three concepts are connected, the first, Value, is used mainly by companies and their products/services. The processing ones, value for customer and perceived value, is focused on the customers and how the value of the product is viewed by the public and the targeted market.

For this project, value is considered the deliverance of data and features to stakeholders in timely matter. More than guarantying releases are made accordingly, it's crucial to keep up with the customer perception of this. In this case, the customer values the deliverance of features correctly, if by any means the feature is incorrect while on production, the customers will not understand how it wasn't identified before it got to them. Making sure errors are detected before release is of major importance. Stakeholders don't get concerned over the expenses caused by the architecture created, for them, being able to decide based on available information is more important.

2.1.3 Value Proposition

BI Development teams will have access to an automatic release process (Continuous delivery and integration pipeline) that encapsulates automatic testing for features being released, automatic integration tests to make sure the release didn't change the previous system status, smoke tests and automatic deploys. To archive this, it's necessary to first create a workflow for version control of the project (database objects and SSIS project) since it's now non-existent.

Currently, BI teams face big problems due to manual and slow releases that are prone to error, this results in low number of releases, meaning, value is not being delivered. Also, version control software for BI projects aren't common, especially for SSIS objects. Teams either communicate well or create dependencies between tasks, e.g. once an object is being changed, it's locked for everybody else. For the database version control, most of the times it's difficult to find a software that fulfils the needs of the team and the project.

The solution proposed, aims to beat the problems identified with the use of a pipeline of CD. With the help of automatic tests, smoke tests, integration testing and automatic deploys and version control over project objects. Resulting in more frequent releases with little time spent on it and less prone to errors.

2.2 CANVAS

Proposed by Alexander Osterwalder, the business model CANVAS is *"A shared language for describing, visualizing, assessing and changing business models"* (Osterwalder et al. 2010). This model allows organizations to describe and think though the business model of the organization, competitors and others enterprises. The following represents the business model designed for the present solution.

<table>
<tr>
<td rowspan="2">Key Partners
BI Teams
Release Engineers</td>
<td>Key Activities
CD/CI Pipeline
SSIS project automatic compilation and deploy
Database and SSIS project version control.
Automatic feature tests
Automatic Deploys
Smokes tests;
Automatic and frequent releases
Automatic deploys</td>
<td rowspan="2">Value Proposition
Will have access to an automatic release process (Continuous delivery and integration pipeline)
Create a workflow for version control of the project (database objects and SSIS project).
Use of a pipeline of Continuous delivery
More frequent releases with little time spent on it and less prone to errors.</td>
<td>Customer Relationships
Support on developing automatic tests
Releases support</td>
<td rowspan="2">Customer Segments
Developers, testers and PO's that aim for more agile releases.
Other BI teams from other companies.</td>
</tr>
<tr>
<td>Key Resources
Jenkins
Octopus
Git
Database Versioning Control Software</td>
<td>Channels
Slack
Email</td>
</tr>
<tr>
<td colspan="3">Cost Structure
Server Machines
Licencing software</td>
<td colspan="2">Revenue Streams
A CD/CI pipeline will offer the team the possibility of more frequent releases, meaning, and more value being delivered to stakeholders.</td>
</tr>
</table>

Value Proposition:

BI Development teams will have access to an automatic release process (Continuous delivery and integration pipeline) and automatic testing to guaranty the release didn't changed the previous system status and automatic deploys. Currently, BI teams face big problems one of which results in low number of releases, meaning, and value is not being delivered. The solution proposed, aims to beat the problems identified with the use of a pipeline of Continuous delivery. With the help of automatic tests, smoke tests, integration testing and automatic deploys and version control over project objects. Resulting in more frequent releases with little time spent on them, meaning value is delivered frequently and feedback is quicker.

Customer Segments:

The main segment found is the BI Teams of the company, all three of them lack a workflow of CD and have problems when releasing new features. The team is composed by engineers with different background such as developers, testers and release engineers. But other external members are considered also customers, these are the product owner, the link between the team and the stakeholders responsible for making sure the right value is delivered, and finally the stakeholders that will their requests delivered more frequently.

Customer Relationships:

The type of relationship that will be stablished with the customers are of support both in releases and in developing new automatic tests for each feature developed.

Channels:

The channels of communication used internally are Slack and Email, these are stablished by the company.

Key Partners:

The development of this solution isn't possible without the help of expertise in the field of releases (release engineer) and without the feedback of the BI teams, making sure the solution found suits their needs. The influence of these partners will leverage the solution to a greater level.

Key Activities:

The activities we aim at do well is a CD pipeline, producing software in short cycles, ensuring that the software can be reliably release at any time. Because CI is a part of CD, version control for every project object and automatic tests are considered activities.

Key Resources:

For this solution, the resources necessary are an automation server (Jenkins), that will serve as an orchestrator, a deploy server (Octopus), SSIS version control software (Git) and another for the data base, not yet chosen. This infrastructure will allow value to be created and delivered successfully, these assets are indispensable to guaranty these and are currently used by other teams in the company for their CD pipeline.

Revenue Streams:

Capturing value in this solution is about delivering features quickly, which will enable stakeholders to give feedback promptly, proving the team with the ability to correct errors of develop new features. A CD pipeline provides a frequent and less stressful delivery of value, which in a BI project means information is made available to stakeholders for them to make decisions based on that information.

Cost Structure:

For a solution of this kind, the main costs are due to software licensing and the need for servers, so it's possible to have different environments.

2.3 Analysing value

Per Verna Allee, "a value network is a web of relationships that generates economic value and other benefits through complex dynamic exchanges between two or more individuals, groups or organizations." (Allee 2002) Any organization that engages in exchanges can be viewed as a value network. It's important to understand how value is created, traditionally it's through the value chain, however, it's a linear model based in the industrial age. But this model doesn't take in count the role of knowledge and value exchange, which for networked enterprise is emerging. A value network generates economic success or other benefits for the participants (Allee 2002). This is possible by having participants converting their expertise and knowledge into deliverables that present value for other members of the network (Allee 2002). Every participant should contribute and receive value, this allows for the success of the value network. Concluding, to build and analyze value, a company should adopt a value network and encourage communication among the participants.

Porter Michael created the concept of value chain, every business has a value chain and it explains how functional the company is and how they manage and communicate with the business units (Porter 1985). If the value chain of a company is too broad, because it ignores important sources of competitive advantages. Being value the subject of the value chain, the previous consists of value activates (distinct activities performed) and margin (the different the total value and the cost of performing value activities), allowing the value chain to display the total value.

Both analyses allow companies to evaluate their value, in a value network, the analysis focus on the communication between participants of the network. This exchange is the definition of value, since individual value will be exchanged. While on the second more formal factors are taken in count when analyzing value, since the object of analysis is the value produced after the effort to produce it was made. For the present solution, the best method to use is the one created by Verna Allee, the concept of value network. The status of the BI teams and their members allow for knowledge and expertise to be exchange between them, leading to a detailed and complete.

To analyze the value of the present solution both approaches have been considered, because this project needs the knowledge and cooperation of different profiles such as release engineers, BI developers and database administrator, the exchange of information creates value, just like described by Verna Allee. But proving it's worth investing in this solution, more formal methods are necessary, understand the effort needed. Above all, any value analysis needs both approaches, one will promote communication and knowledge exchange while the second will prove the effort taken was worth.

3 State of the art

In this chapter state of the art regarding CD on BI projects and this technique reliability is analyzed. It starts with a presentation of the methods and technologies currently used, then, the specific BI projects characteristic, are detailed it by category (database, ETL and general tools). It ends with an approaches which best fit the project and how options were evaluated among themselves.

3.1 State of the art in existing solutions / approaches

As referred on chapter 1, applying CD concepts do a BI Project isn't an easy task, because of that it's necessary to outline components with different behaviors and treat them separately.

To start a CD pipeline, code must be in a version repository, meaning, every object, configuration, data and code most be under the control of a version control software. A typical DW is made of database objects and their data and ETL project. Because there isn't a great amount of information available about solutions that include the two stages, the following section presents the state of the art of CD pipelines for each individual element of a DW.

3.1.1 Continuous Delivery: Database

Unlike code, database version control and deployments are complicated because it isn't a collection a file but a collection of business data (Yaniv Yehuda 2014). Agile techniques have better performance than traditional ones, giving the development approaches available now (Ambler & Sadalage 2006). The goals, when implementing a CD to databases is to have reliability (safe deployments and prevention over hot fixing), visibility (coordination) and agility (automation and teams working together) (Yates 2015). A database isn't just a collection of tables, views and procedures. It is necessary to address other objects such as SQL Server Agents, jobs and others used by the ETL processes (Factor 2014).

Two popular development approaches, for databases, are state driven and migration driven. The first, initially introduced by Redgate (presently used by Microsoft), states that it is necessary to maintain a snapshot of the current of the database structure. To upgrade it to a new version it is necessary to use a tool that will compare the new structure to the current and auto-generate the scripts required (Khorikov 2015). This approach frees the developers from finding the delta of differences and from saving every script used. Migration driven delivery depends on existing scripts, created when developing new features (Khorikov 2015). This scripts should execute successfully at any moment in the database. Table 1 shows the main differences between both approaches considering the main evaluation points.

Table 1 - State driven vs Migration Driven

	State Driven	**Migration Driven**
Merge Conflicts	Straight forward, database is a SQL script	Careful script revision, merging conflicting changes. Error-Prone.
Data Migration	Schemas are objective, but the data is context-dependent, the tool is not capable of generating scripts to handle data	Migration scripts already contain the required code to handle data easily
Workflow	Agile and quick	Manual and slower
Team size/ changes to DB	Big teams with frequent upgrades	Small teams with infrequent upgrades
Control/ Responsibility	Application used	Developers

None of the approaches are infallible, if the approach chosen is for migration, it is necessary to review all the scripts and make a manual merge off all changes, so they don't overlap incorrectly, resulting in unnecessary time consumption. The state approach seems to be the most appropriate, the tool compares the schemas to find the differences with ease, but in table/columns rename cases it is possible that the application eliminates something that should not.

3.1.1.1 Tools

Red gate Database Lifecycle Management (DLM) is an application for database management through every stage from design, architecture, development, delivery and version control in

order to achieve a CD pipeline (Fritchey & Skelton 2015). To accomplish a successful reliable and repeatable release process, DLM has five key activities:

- Versioning, states that every project object should be under version control either it's database objects of others such as ETL packages and data;
- Branching, allows team members to collaborate and develop at the same time;
- Testing, releasing new features requires tests, making sure the previous features aren't affected by the new release (regression tests), if the new features work in the expected way (integration tests) and other tests to check the critical and most important parts of the application;
- Using installation packages, eases the deployment of new features because it creates individual packages that can be used in any environment;
- Automation is key, any task that doesn't need human intervention should be automated from building, testing and deployment of the application.

Red gate follows a state driven approach when deploying databases, focusing on comparing the source and target databases through dynamic scripts bringing both databases to the same state (Fritchey & Skelton 2015). Once the changes are committed by the developer and chosen to be released, DLM creates a diff report between the two databases before they are applied. This guaranties the version control of the object and the creation of deployment scripts.

Flyway is an open-source tool for database version control that uses the migration driven approach, focusing in making migrations easy. (Anon 2017) Unlike other solutions, it has a command line client and plugins add-ons for integrating other applications available on market. Being migration the centerpiece of Flyway workflow, it will scan the filesystem and compare them to migrations that have been applied to the database (Anon 2010).

Another popular tool to have CD in a database is DBMaestro. Unlike DLM or Flyway, DBMaestro uses a hybrid solution known as database enforced change management solution, which combines the version control processes on database objects with the generation of deployment scripts on demand, based on version control repository (Anon 2014). By doing so, DBMaestro ensures database code is covered, the version control repository acts as the source of truth, the deployment scripts are adapted to the targets environment status and handles merges and conflicts with ease (Anon 2014). Figure 3 demonstrates what DBMaestro proposes versus what simple compare and sync offers.

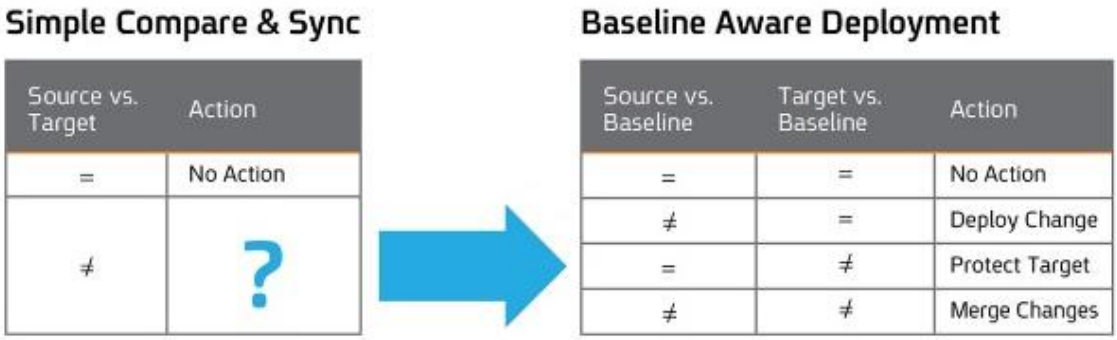


Figure 3 – Comparison between Compare & Sync and DBMaestro [1]

3.1.2 Continuous Delivery: ETL

Feedback should be triggered at every release, it is difficult to archive this after a release on the ETL, as it can be a long process that takes hours to complete. This happens because ETL is associated with data movement between databases (Windows et al. 2014), meaning that some time is needed to deal with this information. At current times, ETL represents more than the traditional purpose, to integrate data into de DW (Hurwitz et al. 2013). Now, ETL processes data in different ways like cleansing, profiling, auditing and provide big data extraction. This new technology aims to beat the major problem in a traditional ETL, latency because of the data volume. Big data enables organization to manipulate vast amounts of data at the right speed and time (Hurwitz et al. 2013).

3.1.2.1 Tools

IBM Information Server is a market-leading data integration platform from IBM that provides massive parallel processing, delivering highly scalable and flexible integration to handle a variety of data volumes (IBM 2016). It's architecture provides a repository where information about data sources, integration jobs, data warehouse and reports are located (Ballard 2010a). The repository acts as a version control tool because the metadata is saved and uploaded to the repository server, accessible to every user. This approach provides advantages to the organization, promoting collaboration across developers and business users. In thought throw releases, work is structured by folder distinguishing major releases from incremental fixes (Ballard 2010b). Deployment packages are created from the repository server, where the completed features are included. Each release candidate has a dedicated project, it can include single releases, incremental fixes or other objects. Each of them is deployed to the server, everything is achieved with the use of Information Server.

Talend, is an open source data integration software that runs natively in Hadoop and it is the first data integration platform to give users great performance (Anon 2016b) It is especially useful for big data projects. Just like the IBM solution, Talend offers its users with version

[1] Source: The Definitive Guide to Database Version Control (Alex Papadimoulis 2012)

control, enabling collaboration between team members in a shared repository (Anon 2015). Deploy is made through graphical console, activity monitoring and service locator capabilities.

Regarding the tool currently used by the organization, SQL Server Integration Services (SSIS) is another platform for data integration by Microsoft. Easy to use and intuitive because of the components available and options available on them. The ETL process is difficult to version, comparing to traditional code versioning. The project is nearly impossible to version since it can't be merged using, e.g., Git and SSIS doesn't offer this option natively. This happens because the files are XML and have different identification tags, when opened with another computer the identification tags can change. Merging situations like these, is time consuming and ultimately of high costs to the organization, most of them adopt the method of one developer per file (Costa 2015). But this goes against the CD principals, unfortunately most tools that try to merge ETL projects fail, so this is the easiest and less costly choice. Communication is key, to guaranty no developer works in the same object as another. Deployment is eased by the tool that offers possibility to deploy projects via command line. Proving the developers with the ability to create custom deployment scripts.

These are just three examples of leading tools for data integration, both IBM and Talend offer the ability to version control ETL projects and, on IBM case, to use a third party together with IBM deployment service. Talend has great features for ETL processes especially for big data, currently a big trend on BI projects. Finally, SSIS is the old and trust worthy tool that offers a big quantity of tasks with easy configuration. Right now, SSIS lacks the appeal of the other two feature wise (Anon 2016a).

3.1.3 Continuous Delivery: Database / ETL General Tools

Apart from the previous described tools, others, of more general use, are considered since respect both Database and ETL needs when versioning objects, build and deployment.

Starting with Git, which is an open-source version control tool whose focus is on performance and integrity. In Git, data is store as a stream of snapshots instead of a list of file-based, making it a mini filesystem. In addition, operations are local, making most operations almost instantaneous since the entire history of the project is on the local disk, once changes are ready to commit they are pushed into the online repository. The two previous facts justify the performance point, integrity is guaranteed using unique tokens given to each commit, only adding information to Git database making sure previous information isn't lost and finally, on Git files go through three stages committed (data is stored in a local database), modified (changes were made but not committed on local database) and staged (files marked to be committed next). Finally, Git promotes branching, a non-linear development method. This feature is fast and easy to manage in the long run. (Chacon 2009)

Team Foundation Server (TFS) is a Microsoft application for version control, agile development and data collection and reporting. Unlike Git, TFS uses a centralized version control system, where teams checks in all their work into team foundation server. (Minium

2006). Also unlike Git, TFS presents others features such as project management, work item tracking (used to track bugs, requirements, scenarios and tasks), team build (producing public builds using MSBuild) and data collection and reporting (allowing analysis about bug tends, test coverage among others). This tool too is an automation server for automatic build and deployment.

Concerning other automatic build and deploy, two of the most popular tools are Jenkins and TeamCity. Jenkins is an open-source CI application used to automate a range a task such as building, testing and deployment. Its versatility and use of external components is aim at archiving different goals. Jenkins has four major features that distinguish it from others (Kphsuke Kawaguchi 2017):

- Easy installation, with just on command it is possible to install Jenkins;
- Easy configuration thanks to its friendly interface;
- Plugin ecosystem, Jenkins integrates with build and supply chain management tools, proving teams with the possibility of fulfilling their needs with little problem;
- Distributed builds, Jenkins can distribute stages to multiple computers, having compatibility between operations systems.

Jenkins is an example of success, big companies use Jenkins as their CD tool, the plugins, community behind it and pipeline structure make Jenkins a go to application to satisfy different needs.

TeamCity is a java-based CI application, just like Jenkins, it uses plugins to improve team collaboration. Its key features are, different from Jenkins are:

- Efficient build management, for maximum efficiency TeamCity used a build grid to run different build configurations;
- Code quality maintenance with the use of server-side code analysis features, searching for code duplicates and performing code coverage, freeing up developer's local computers resources;

Table 2 summarize the differences between Jenkins and TeamCity considering import factors and features common to both.

Table 2 – Difference between Jenkins and TeamCity

	Jenkins	**TeamCity**
Ease of Use	Simple, requires some knowledge	Intuitive, easy to learn

Community Support	Has bigger community, since it has been around for longer	Recent tool, lacks a large community with different profiles
Pricing	Open-source, free	Enterprise licences can be bought, free version only allows 20 builds and 3
API	Greater extensibility	Less information available
External Integrations	Great number of free plugins	Reduced number of plugins

3.1.4 Continuous Delivery: Deployment

Octopus Deploy is an automated deployment server. It makes it easy to automate deployment into development, test and production environments. (Octopus 2017). Octopus uses NuGet packages as storage mechanism, which is a package manager designed for the Microsoft development platform. The packages are generated after developers commit their code into the existing source control system. This code is later compiled by the CI server and then packaged, ready to deploy. Because Octopus deploys packages to environments using various steps and custom Powershell scripts (Octopus 2017), it gives teams a great advantage and adaptability when deploying any project regardless of the type. Using Nuget packages isolates releases so, it is impossible to tamper with files inside the release package.

Because the company has a series of different projects, which use different technologies, Octopus has proven to be the best choice because of its versatility and ease of use when deploying software into various environments and servers.

3.2 Solutions/approaches identified

Although TFS is currently used, migration to Git is the path being followed by the organization. This decision affects every technological team but there is a possibility of using another of the investigated tools, after stabilization of Git usage with every team.

Regarding the orchestrator/build software, the organization uses Jenkins. To maintain homogeneity this is the best choice for the present project.

3.3 Evaluation of existing solutions/approaches

The applications chosen have to fulfil the needs of both BI teams and the organization. To evaluate the solutions, it's necessary to consult other tech teams on the organization, to keep

processes dependency free and outline the needs of BI teams to understand what software will respond to these.

3.4 Conclusion

After the state of the art study, the adoption of CD was proven to be a success factor in any used release process due to the use of pipelines, which provide several staging areas to test the releases, making sure that the final deploy into production will be successful and cause no damage to the current version.

Applied to BI, it will improve the release quality as well as the release frequency, since developers are uncomfortable releasing new features manually and directly in production. Currently the organization uses CD in several projects and to keep homogeneity through all of them, the goal is to use the same tools for CD used in the other projects, Jenkins for orchestrating the pipeline, Octopus Deploy for deploying software into the environments and finally GIT for version control.

4 Solution design

The solution chapter aims to expose the solution design as well as explaining the decisions made. It starts by listing, functional and non-functional requirements. Then, there is a detailed architecture design, describing the different business processes, using several UML diagrams. The chapter ends with the presentation of the alternatives discovered during the design process which are considered relevant for the future.

Concerning the most common problems in a BI project, the solution should take into consideration the following issues:

- **Data handling**, most database releases imply data manipulation/change that can be easily scripted but hard to keep under version control, since updates affect data and not objects. Consequently this scripts need to be re-runnable (able to run at any time and produce the same effect without putting other data at risk) and saved in the repository in an explicit folder.

- **Operational data is becoming harder to access**, to keep the source database isolated from ETL changes, every necessary operational data is replicated onto database. Until now this was the stipulated process, however, to keep data isolated and not replicated (due to security issues) and because of architectural changes (with the goal of designing the application as a service) data streaming is now event based. Meaning instead of transferring data in batch, data is transferred by messages (Event-Driven Messaging). Event-Driven Messaging is a SOAP design pattern that aims to address the inefficiencies related to the traditional pooling based model (Frank & Zeng 2013). This allows the application to communicate with other application via services, this changes lead to the necessary of rethinking the current process.

- **Infrastructure needs**, data growth and performance are issues that impact and influence the behavior of the DW, reporting, analysis and the ETL process. Larges amount of data conditions DW access as so neglected features development that don't regard infrastructure needs and don't follow state-of-the-art

technologies/techniques. DW volume is expected to grow quickly, with the integration of offline data and clickstream, even though physical changes can be made to gain advantages these shouldn't be the first option. Team members should investigate new technologies and techniques to resolve the problems faced. After migrating everything to a cloud service, big data is now a reality in the organization, the current project needs to adapt to this and improve its architecture. Data Lake is "a storage repository that holds a vast amount of raw data in its native format", in which, "data structures and business requirements do not have to be defined until the data is needed" (Harris 2016). At the organization, the data lake will be an ecosystem of all the technologies where users can access data; this was one of the results of an ongoing investigation (which aims to remove data synchronization of data in between servers and have it available in one single infrastructure), of what was most fitting for organization user's necessities.

- **General Data Protection Regulation (GDPR)** is a regulation for personal data, recently released by European Union, for all European individuals. Because this regulation affects every organization, either processor or controllers and regardless of the local it takes place, the organization needs to conform to this rules. Being a data driven organization, a few important measures need to be taken into account regarding subject data:

    - Organizations should clearly and easily make available the terms and conditions, so users can provide consent or withdraw;

    - Any data breach must be made public within 72 hours;

    - Users have the right to access data concerning them, even requesting for personal use or to be transferred over other organization;

    - Users have the right to be forgotten, meaning the organization must erase his/her personal data, cease further dissemination of the data and potentially have third parties halt processing of the data.;

    - Organizations should implement privacy by design, regarding data protection. Appropriate technical and organizational measures should be used in order to meet the requirements of this regulation, protecting the rights of the users;

    - Lastly, every organization needs a Data Protection Officer (DPO), someone with understand of the regulation, a clear and capable communicator and able to effectively she his knowledge.

4.1 Proposed Architecture

4.1.1 Architecture

The architecture is the central artefact, it should illustrate the system so that it can meet the requirements raised during the analysis, both functional and non-functional requirements. Non-functional requirements represent aspects relevant to the quality and reliability of the solution, while functional requirements represent the necessities to be developed and use cases for those. In a CD project the functional requirements follow the principles of repeatability, reliability, automation, versioning, frequency, communication and quality. Releasing software should be as simple as pressing a button (Humble et al. 2010). Non-functional requirements are performance, security and usability.

According to Martin Fowler, David Farley and Jez Humble, there are seven principles for software delivery (Humble et al. 2010):

- Repeatable and reliable releases: the release process should be easy because every part of it is tested several times before, to achieve this, everything possible should be automated from building to releasing. If well done, version control can provide the ability to fully automate the deployment process;

- Automating almost everything: automation isn't the solution for everything, user acceptance tests rely on user experience and therefore can't be automated, such as requirements that need human approval. But this is just a small part of the release process that shouldn't block the release, but take advantage from the automation of everything else. The first things to be automated should be the bottlenecks identified to being with: build, deploy, test and release. Gradually, everything should be automated;

- Version Control: everything that is a part of the software should be kept under version control. Apart from keeping a history of changes, version control allows software to be installed at any workstation and to see the build version of difference environments;

- Frequency is key: releasing big chunks of code can be hard, so integrating and delivering value frequently, e.g. per feature, is an agile and easy way to deliver value quickly and with less problems than before;

- Building quality software: delivering value is important but making sure features have quality, tests results shouldn't be ignored and fixed later after released;

- Discipline and communication are indispensable, CD process give power and independency to teams, giving them the ownership of the release process. This means

teams must be responsible when establishing what a finished feature means and what their mission is as a team;

- Continuous improvement, the release process developed initially will not serve the teams needs in the future, just like development techniques, the release process should evolve. Teams should discuss what needs to be improved and find ideas to improve things.

The design pipeline architecture (Figure 3) should represent these values while making sure performance, security and usability isn't a setback. When dealing with corporate information security is essential and can't be overlooked.

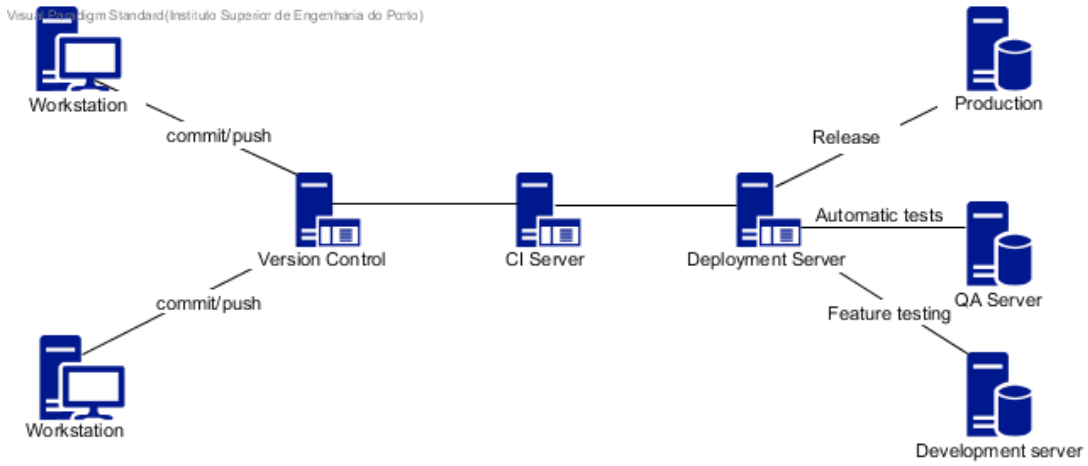


Figure 4 - Concept Architecture

Considering the requirements raised, the following architecture concept as illustrated on Figure 4 is intended to simplify the explanation of the pipeline architecture. Figure 5 presents a more complex business model to represent the pipeline workflow.

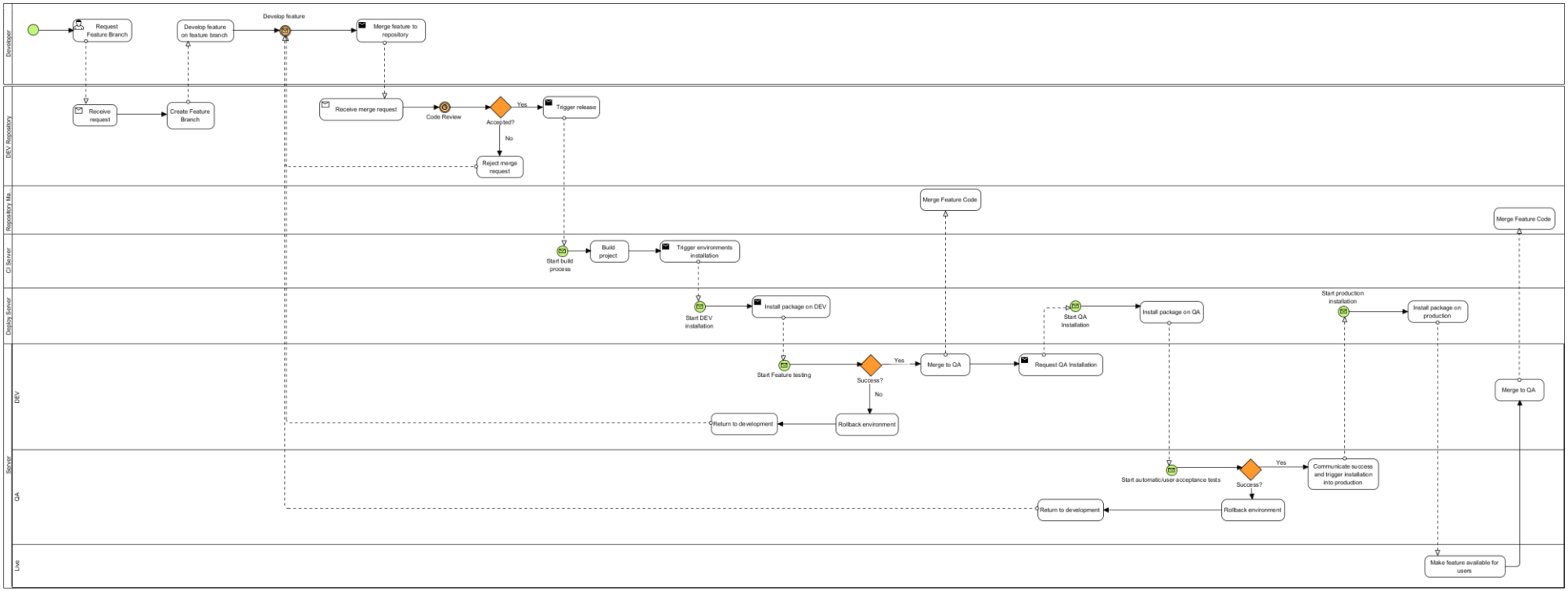


Figure 5 – Release pipeline workflow

As presented on Figure 5, the pipeline workflow starts with the creation of a feature branch from the integration branch, unlike a master branch, where the code of the application is located, the integration branch is where every feature branch is integrated before going to the main branch. To merge to the integration branch, the developed code must be reviewed by other developer, this is known by merge request. If the feature is accepted, the CI server will be triggered to create and build call the deployment server to install the feature in the development environment. In here, the feature goes through the stages of feature testing and owner approval. Once it is approved, the developer proceeds with the merge to the testing branch, a replica of master but where new developments were first merged. This will trigger the CI server once again to proceed with the installation of the feature in the QA (Quality Assurance) environment, where automatic feature tests, regression tests and user acceptance tests will run. QA environment is synchronized with production environment, meaning tests will run in a similar setting as production. Only if tests are well succeeded in QA, the package will be finally installed in production and then merged to master.

4.1.2 Detailed Architecture

4.1.2.1 Components

A component diagram doesn't describe the functionality of the system but it describes the components used to make those functionalities work. Figure 6 shows the seven components identified, the development server, version control tool, build server, deployment server, development machine, QA live and production. The developer machine communicates with the version control to commit and push changes. The build server accesses the repository to get build the code that communicates with the deployment server, which installs the packages onto development, QA and production environments.

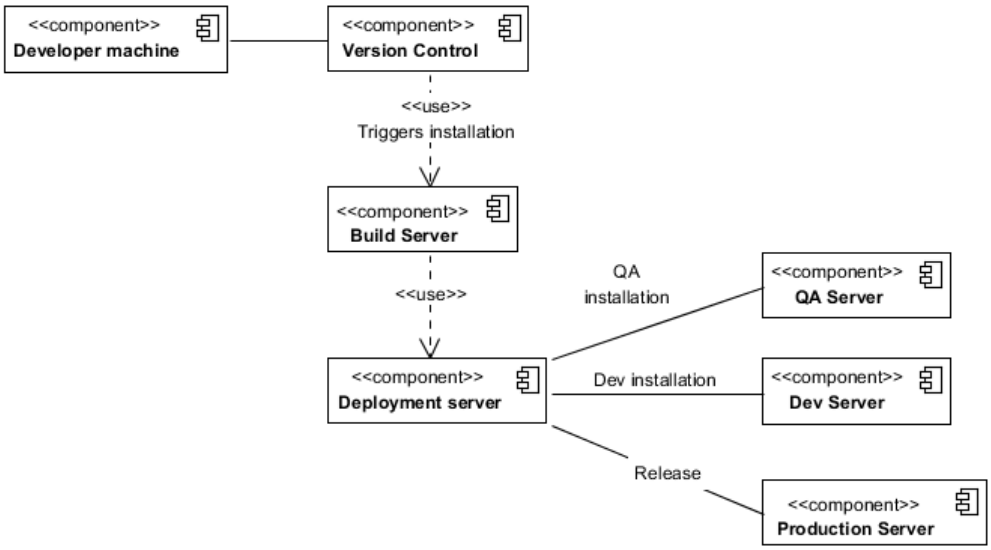


Figure 6 – UML Component Diagram

The main issue with this approach is the refresh of data on every environment. This happens because DW systems have a heavy volume of data, having access to the hard drivers, is easier for the organization to manage it, but since the organization uses cloud services, backing up or copying a database is a complex and time consuming task.

4.1.2.2 Data Flow

During the process, the feature will go through two major processes committing/pushing and installation in other environments. If a bug is found on production, a new branch, denominated as hot fix, that instead of branching from the development repository, branches from master since it needs to be an exact copy of code running in production. Figure 7 shows the flow of information through the pipeline, depicting the system requirements graphically.

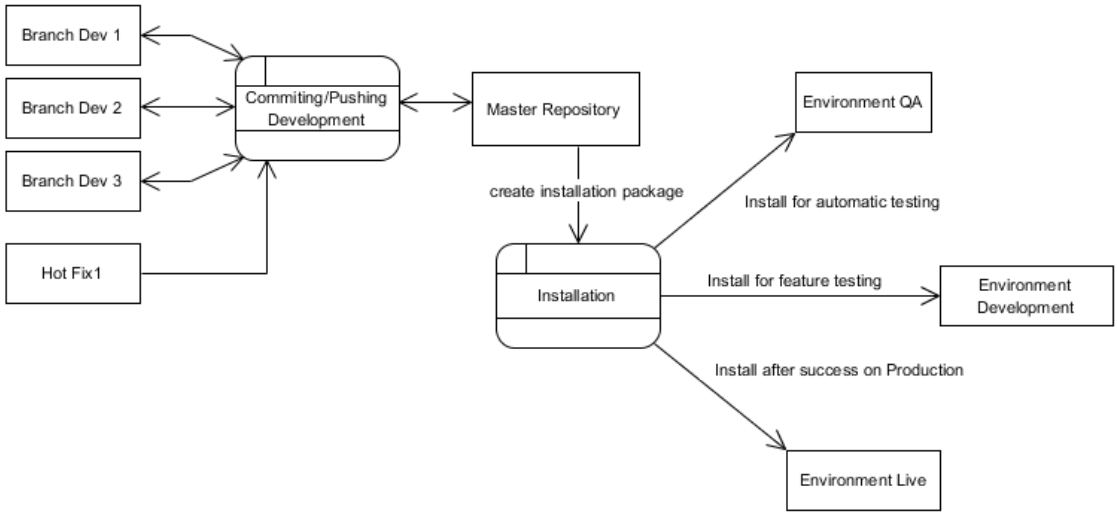


Figure 7 – Data Flow Diagram

4.1.2.3 Sequence and Activities

A feature lifecycle has 11 stages: branching, development, reviewing, testing, owner approval, merging to integration branch and master, build on CI server, deploy on server, automatic tests on QA server and finally, if everything is successful, release. Figure 8 illustrates the previous, only a success scenario happens at every stage.

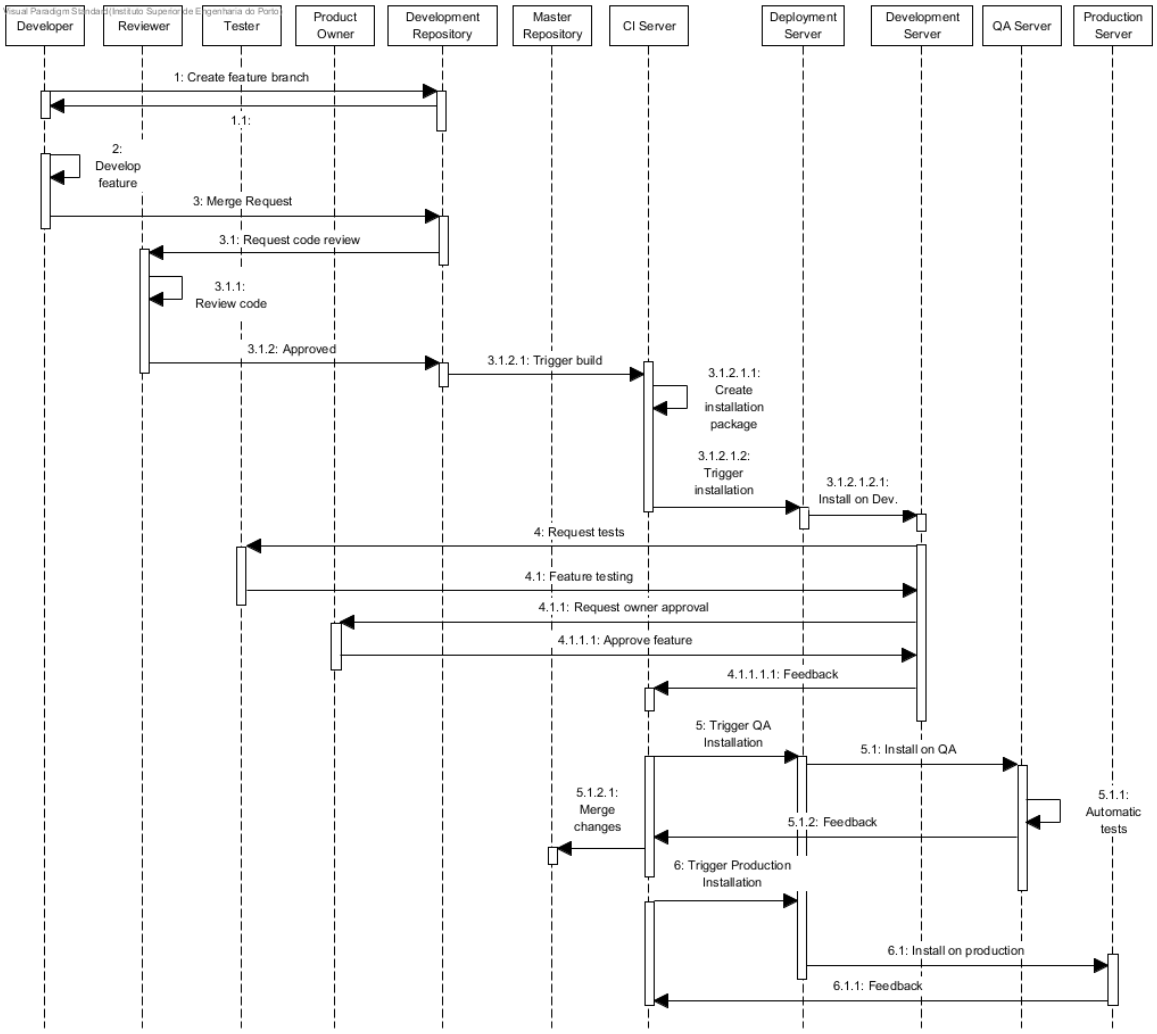


Figure 8 – UML Sequence diagram of feature lifecycle

A feature enters the pipeline of CI when the developer creates a branch base on the integration branch and develops the feature on it. When the feature is ready, the developer requests to merge the feature code into the integration branch, the request is reviewed by another developer, making sure development guidelines are being followed. If approved, the feature is merged into the integration branch and the CI server is trigged to build the project and start the deployment on the development server. The following stage is testing, where the tester accesses the new feature knowing the acceptance criteria. The feedback will trigger the next stage, which is owner approval, or return to development. Subsequently, in case of positive feedback, the feature is ready to be release, but first needs to pass automatic acceptance tests, regression tests, user acceptance that will run on the QA server. After all of them test positive the feature is released. Automatic acceptance tests and the acceptance criteria should be defined by the developer. Regression tests make sure the system as kept its stability and other features wasn't altered. Finally, user acceptance tests validate the user's requests are correctly executed.

Figure 9 displays the possible scenarios and workflows for each decision point.

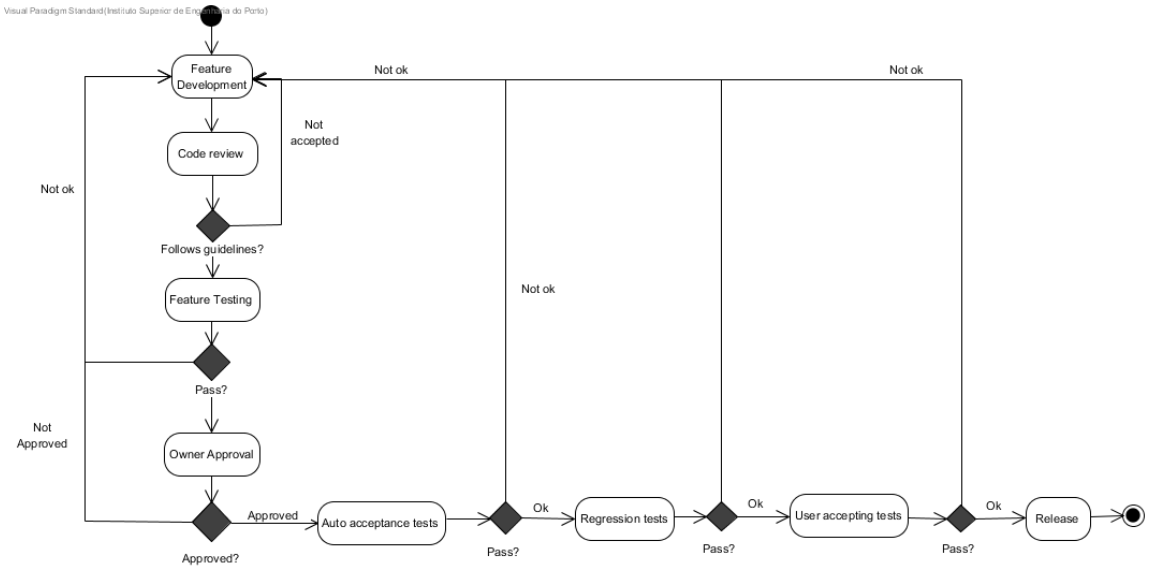


Figure 9 – UML activity Diagram of pipeline workflow

As for the repository branches activity sequence, Figure 10 illustrates how the branches interact with the integration branch and master. The first work as a security measure so that master is a representation of the current state of the application.

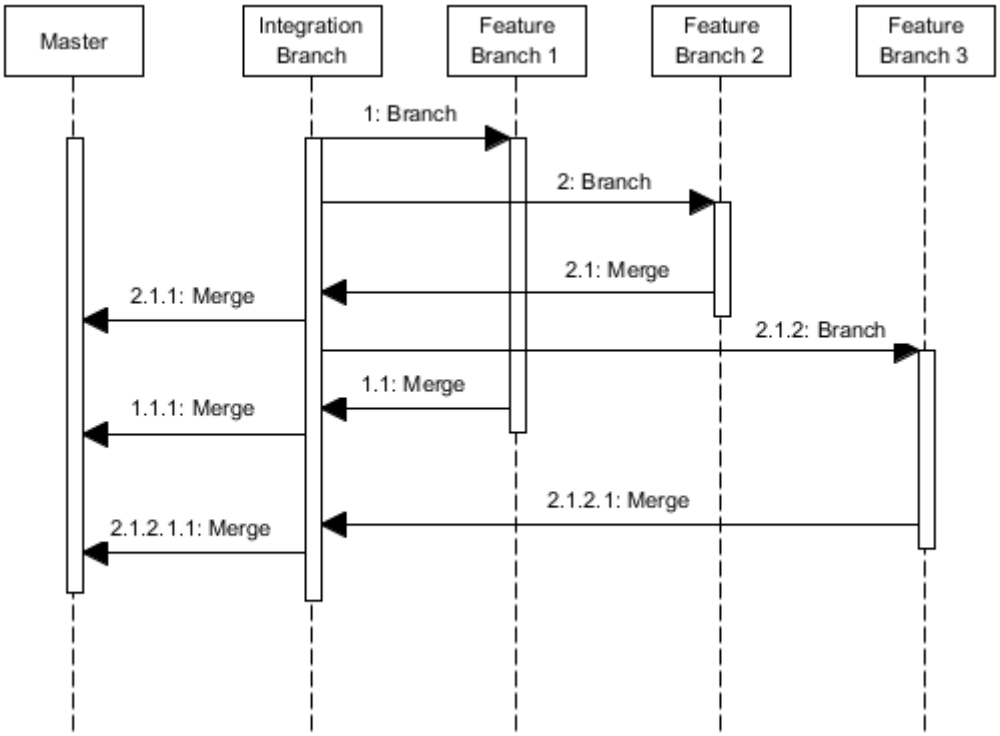


Figure 10 - Repository workflow

4.1.3 Alternatives

During research an alternative architecture was found, one that takes into account the organization infrastructure and the requirements. The major arguments that lead to this alternative are the costs associated with more application servers, needed in the first solution, and ease of use of Git by the teams, presently teams don't use Git or any other version control software.

The first argument was raised due to the number of application server's necessary for testing releases. In this case, as illustrated on Figure 11, in this architecture, only one testing environment would be used for automatic tests, feature testing would occur on the developer machine instead of a dedicated server. But this approach requires more licenses for database instances since every computer would be considered an isolated database. This method also presents itself with security issues since the databases can't have organization information such as sales, this is due to regulations regarding user data exposure.

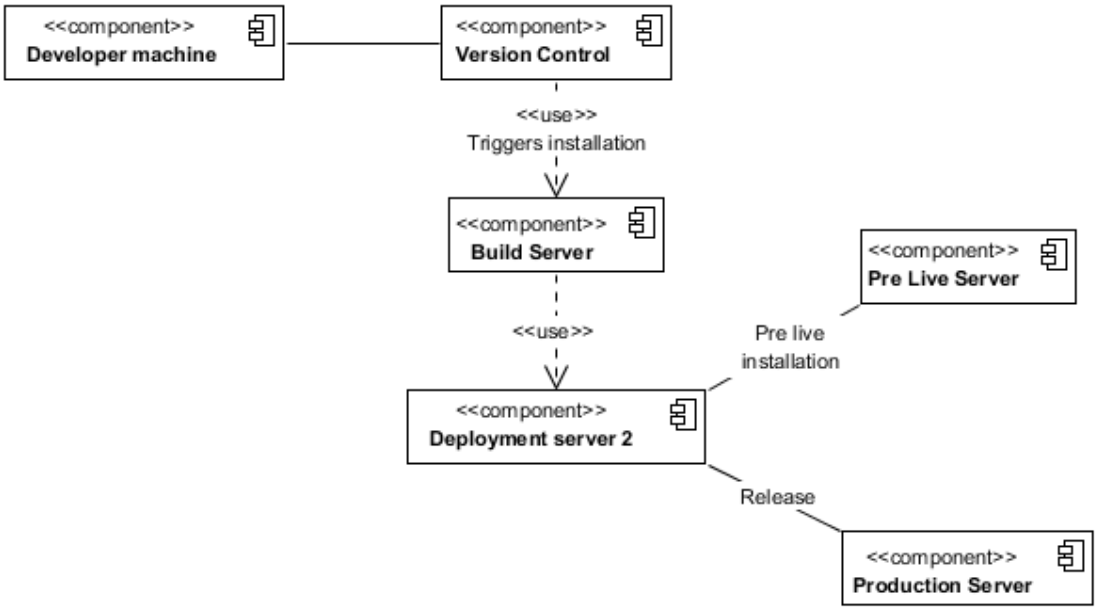


Figure 11 - Alternative architecture components

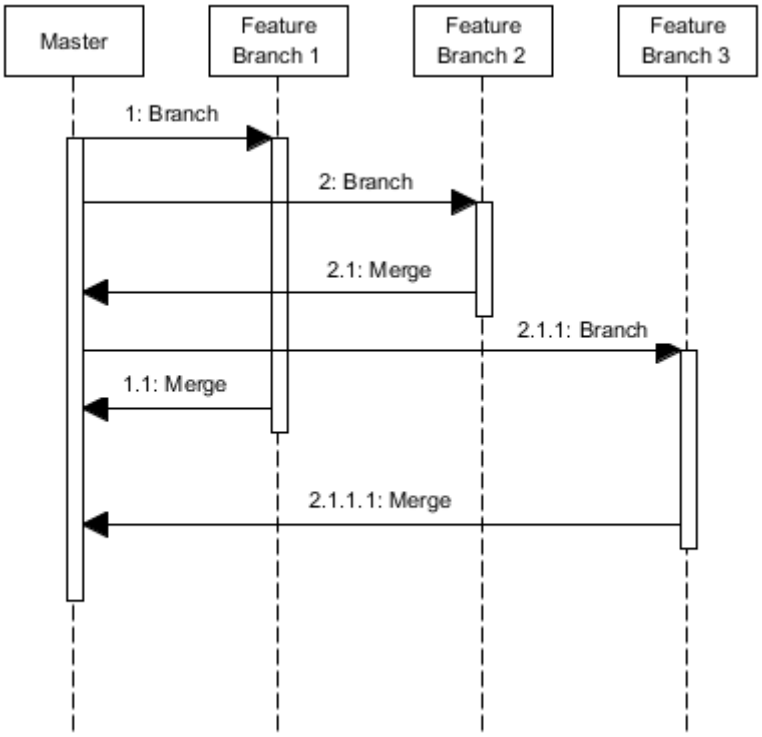


Figure 12 – Alternative repository workflow

Regarding the repository workflow, things would also change, as showed on Figure 12. For this architecture, only one master repository is considered eliminating the use of an integration branch. Although this removes the safe keeping, the integration branch provides it, it is easy the use of feature branching for the developers and removes one process during the pipeline, the merging of the feature installed in the master branch.

4.2 Final Architecture

After thoughtful consideration over the proposed solutions, it was concluded that the first one presents more added value in comparison with the alternative. All the features will be developed in a branch that later will be installed on development, QA and production environments.

Figure 13 - Final Architecture

As stated on subchapter 3.1.3 Continuous Delivery: Database / ETL General Tools, the tool chosen for orchestration CD tasks is Jenkins. It will be responsible for building the solution and

triggering Octopus when installing the release package into other environments. At this point in time, Jenkins is used for the majority of projects in the organization but the plan is to start using TeamCity and thrust aside Jenkins over time. For the time being, while the engineering team is performing tests to TeamCity, new projects are developed using Jenkins.

As for the deployment software used, Octopus is used throughout the company and therefore is to be used in any new projects, to keep consistency through the organization and ease the maintenance.

4.2.1 Setting up git repositories

The first stage of this project is to setup Git repositories for each team project. The easiest and most everyday way to version software is to do it in small, less complex parts instead of one big project. The current ETL project consists on an intrinsic set of processes responsible for different products, the first step into versioning it is to separate it in several distinct projects that represent each process/product. Most features are developed only for one product, but not rarely one feature might change more than one project. Regarding the first scenario, having projects separated by product will help teams remove dependencies from the main project and other processes, therefore, releases can happen freely without harming other developments or products. In the latter scenario, even though multiple projects will be affected, a number of release packages will be created for each changed project and then installed.

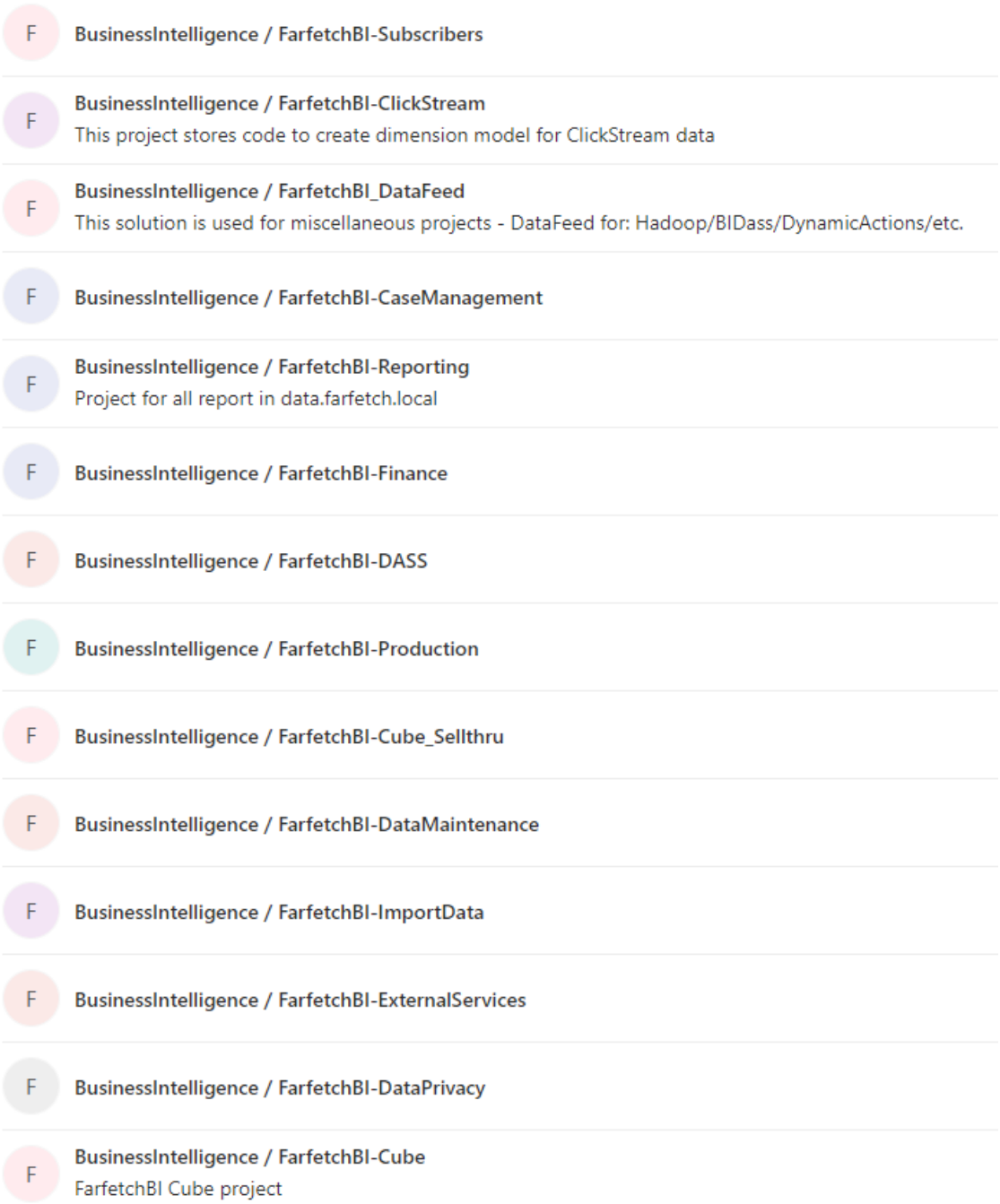


Figure 14 - Some of the CI repositories created

4.2.2 Setting up pipeline on Jenkins

The Jenkins pipeline has four stages, starting off with the release build, making sure the release package has no build errors. Before this stage the feature is developed in a devlopment environment, when the development is comple the developer creates a release package which will be installed again making sure the release package was created

successuflly. At this stage, feature testing starts, where another team member will test the previous development. After the development is tested and approved by the product owner, the next stage is triggered, the release package is installed in a quality secure environment. This environment is a replica of production, where releases are fully tested to make sure it encounters no errors when releasing the feature to production. In this stage, automatic tests will also run. Finally, the release package is installed in production.

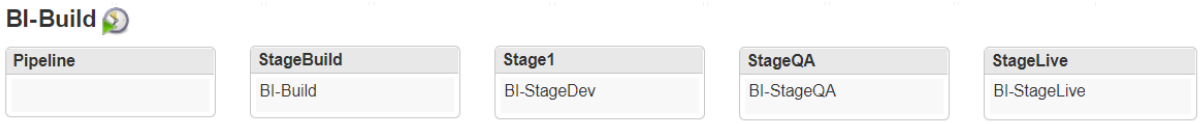


Figure 15 - Defined pipeline on Jenkins

The last stage is manually triggered, as shown on Figure 16, after the success of all previous stages, the pipeline stops and waits for the confirmation to complete the last step.

Figure 16 - Pipeline after sucess of the first three stages

Finally, when every stage has run successfully the pipeline execution is complete and the release package deployed to production, as presented in Figure 17. In the last three stages, Octopus is called, executing its pipeline.

Figure 17 - Pipeline after every stage is successful

4.2.3 Setting up pipeline on Octopus

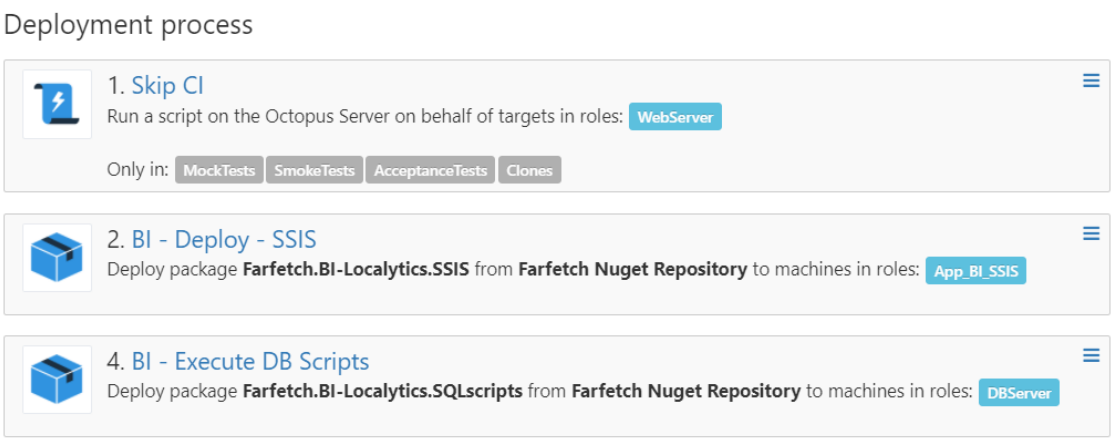


Figure 18 - Octupus pipeline

The Octopus pipeline has three stages, CI, SSIS deploy and database scripts execution. Into the CI stage, each deploy is performed in an CI server where automatic tests will run, by the end, depending on the tests run successfully or not, the server is rolled back to its previous state and a response is sent to the pipeline triggering the next stage (in case of success) or not (if tests aren't successful). Currently, BI doesn't have any CI tests running, but the goal is to use the existing automatic tests and the feature tests developed during testing. Next, the ETL packages for release are deployed into the server, for which the current pipeline is running. Finally, in this final stage, database scripts are run against the server database.

4.3 Conclusion

With the all requirements defined, the solution design phase started with the designing of a possible server architecture and the flow of states for each software development. Both were designed with the teams help to make sure the solution was adequate to their needs and therefore, make sure it will be used. This solution is modelled by other ongoing projects at the organization, its goal is to use an orchestrator (Jenkins) as the pipeline stages manager and the use of Octopus Deploy for the new software deployment.

The solution implementation was divided by stages and projects. The main ETL project was divided in several small projects, independent from each other, i.e., each project has its own repository, reducing dependency between projects. The development in stages will allow each stage to be developed separately in the respective timings not compromising team commitments.

5 Evaluation

This chapter purpose is to assess project evaluation, addressing objectives measures taken into account, indicator, evaluation and support of hypotheses as well as their tests.

5.1 Solution Evaluation

5.1.1 Assessment Variables

The experimentation lifecycle consists of exploration, hypothesis construction, experiments, data analysis and conclusions. There are several challenges associated with evaluating a production system because of limitations on directly gathering important information about it (Stricker et al. 2017). Carefully selected key performance indicators (KPI) identify precisely where to take actions to improve performance. (Weber & Thomas 2005) KPIs focus on critical aspects for the organization, which will possibly lead to success in the future (Weber & Thomas 2005). These indicators should be monitored constantly, depending on the necessity and relevancy of its results.

In CD projects, the indicators evaluate mainly time, number of successful releases and satisfaction. The success of a pipeline is difficult to measure; a successful installation doesn't mean the feature release is effective. In this case, performance is of major importance so time indicators will be measure and it needs to be measured correctly. Also, because the development of this pipeline is meant to facilitate the release process to developers and improve release frequency, satisfaction inquiries will be distributed among developers and stakeholders. Finally, it's important to understand the number of features released, bugs detected among others, so quantity indicators will be also considered when assessing variables.

When releasing new features on a DW there are several tasks that need to be taken into consideration. The database changes made during development should be reflected in

production, this is archived by running the scripts previously before and executing them in production, and next, if necessary for the installation, the ETL should be updated and deployed, directly to production. The ETL project packages can't be merged using a specific tool but rather a complete replacement of the previous version with a new one. The greatest impact of using CD in this environment is the control over each release version, making possible to easily integrate new features and to develop several features at the same time without merging problems when releasing them, with the use of CI. Even though ETL packages can't be correctly merged using a tool, CI will warn developers that new changes occurred in the package they are currently working on, making it easier to merge changes manually and keeping up with them.

Another reason CD will improve the release process is the installation in several environments along with feature testing, before releasing it into production. This offers stability to both, the software and the team, as new releases are tested before installation in production. With the current release process, errors due to releases can only be detected in the next ETL run, so, if the release breaks the ETL, the nightly process will be delayed, and therefore, the databases will only be available to users later than usual. With CD it will be possible to test each release before deploying it to production, preventing the ETL from breaking and provide users with data at the expected time.

5.1.2 Supporting Hypothesis

A CD solution will improve the release process as well as the quality of software delivered, because each release is tested several times before the final release. Because the release process has several factors to be considered, measuring the success of this hypothesis can be challenging. To start off the major problems with the current solution need to be gathered, these will be the first to be tested in the new solution, the next step is to create key performance indicators that prove the hypothesis.

In the current procedure for a release, after the feature it's ready for release, the developer runs every database script on production and replaces and deploys the new package in the ETL project. This solution faces several challenges, being the most interesting the time to production, manual releases and errors due to releases. The latter problems can be reduced by using CD, since the release is automated and new features are tested in appropriated environments before going to production. Regarding the first one, time to production depends on the ability to quickly test the new feature on other environments and this variable depends on the existing software and possibility of having a reduced set of data to test new features.

As expressed before, the new solution will address the problems described above and its results for the following key performance indicators (KPI) will show an improvement from the current solution. The KPI's were defined with the help of the BI teams within the organization, according to their needs, are the following:

- **Time to production** – It starts when the feature is ready for development and ends when it is in production through CD pipeline. Time to install a feature into production is one of the main problems with the current solution, releases can take from minutes to hours, possibly leading to a large downtime. To the company, teams and stakeholders this is a key indicator of success for the new release process because it will lead to more frequent releases. Because time to release is such an important performance indicator, measuring it can be a challenge. According to agile development a story/task includes the development, code review, tests and release, so, the release starts after the product owner approves the story and ends at the moment the feature is installed in production. With the help of the pipeline and issue tracking software used by the organization, it's possible to determine the time each release takes.

- **Number of bugs that escaped the stages of CD** – The goal of the pipeline is to minimize bugs on production this is possible thanks to the multiple stages of testing. This indicator shows the number of bugs undetected during the pipeline, which are on production. Currently, post-release can lead to errors because of the lack of environments for testing, features propagate from development to production therefore it is possible that new releases break the software in production. With CD, bugs will be detected in earlier stages, preventing post-release errors. But the process is not infallible, it is necessary to identify the number of bugs that are not detected in the various phases of CD and how they can be detectable.

- **Total regression test time** – Track the total time it takes to do a full regression test. Regression tests checks if the system respects the previous requirements by running a set of automatic feature tests. This is mandatory, because it is important to do a full regression test prior to any production deploy. Not doing so, it introduces risk to the deployment. The purpose of CD is to provide the results of each phase as quickly as possible, giving the team the ability to correct faulty builds more quickly to deploy the release forward, once again, as soon as possible.

- **Time to fix broken build** – if the pipeline fails during build, it's important to fix it quickly to not block other releases. It's important the pipeline is always free and faulty builds are corrected in the least amount of time possible so the pipeline is free for other releases to go forward. If the correction takes a long time, it is recommended to roll back the software to its previous version, freeing the pipeline for further releases.

- **Bug resolution time** - teams should fix bugs as fast as they appear, this stands side by side with code quality. A successful CD pipeline requires quality code, this mean commitment of the team. Code quality together with multiple stages of testing will allow the number bugs to be low but once they are found teams should be able to fix it quickly. This allows the organization to understand if the pipeline is quick easy to use.

- **Production downtime during deployment** – There is a cost to production downtime, either in satisfaction from stakeholders or financial infrastructure costs. Because the BI platform is a decision support system, its availability is crucial for the users. Data should be accessible whenever the users need it. During releases the system can be unavailable but with CD, downtime should be minimal and always checked for improvements.

5.1.3 Evaluation Methodology

Other major issue with pipeline evaluation is how is measured. Like previously mentioned, the goal of the pipeline is to provide easier and quicker releases for developers, while, at the same time, delivering features, to stakeholders, quickly. Thus, satisfaction inquiries will be used for both stakeholders and BI teams, allowing users to express themselves. This will provide the team with possible improvements and other problems that otherwise wouldn't be considered. Because of the sparse location of stakeholders and the nature of a decision support system, where consultation is the primary task, the inquiries show themselves as most appropriate form of evaluation, since it will allow evaluating the opinion of the stakeholders on the effects of CD in their daily tasks before and after its implementation. This will allow confirming the influence that a CD pipeline will have on the stakeholders, their experience with the software and their satisfaction with it. The inquiries need to consider time to request fulfillment, software performance, number of bugs detected and downtime experienced during and after release. Besides the previous points, the inquiries also need to measure success for more specific points such as release time, ease of use and success on dealing with conflicts. Each of this issues should be evaluated in a scale from -2 to 2, where -2 means worsened exponentially, 0 means nothing changed and 2 means improved significantly.

The satisfaction inquiries will have seven evaluation points divided in two sections; the first four apply to stakeholders and reflect the user experience of stakeholders with the software instead of indicators. The last three apply only to the BI teams since they are related to technical factors and the impact of CD on development:

- **Time to request fulfillment**, the main setback for stakeholders when requesting new features is the time it takes to be release, this is due to the current complexity of the process release;

- **Software performance**, user experience is an important issue in decision support systems, information needs to be extracted quickly and major reports need to be fast to export. Therefore, it is intended that users evaluate the performance of the software. The goal of having CD is to improve releases and promote, not only new features but also technical tasks such as improving procedures performance;

- **Number of bugs detected**, using CD allows each feature to be tested several times before the last installation in production, which means errors and bugs are detected earlier but some might escape thought the test battery. Stakeholders suffer the most

with bugs in data, it is important to make sure users see an improvement regarding bugs detected after release in production when everything is release using CD;

- **Downtime experienced during and after release**, with the current release process releases may lead to big downtimes, either because a rollback was necessary or because releases cause critical errors. This is crucial for stakeholders, decision support systems need to be available at all moments. So, reducing downtime is important and with the use of CD the goal is to reduce the time and improve the user experience;

- **Release time**, currently release times are irregular and don't provide its users with an accurate time for availability after releases. With a release pipeline, it will be possible to correctly predict the time necessary to software availability. With this, it will be possible to prove the success of the pipeline and the improvement in the release process after the use of it;

- **Ease of use**, manual releases require developers to modify several objects, run a sequence of scripts, test and deploy the release by themselves. It is assumed that all changes made during development are registered, making it possible for developers to follow the release step by step, in the long run this process is not easy to follow and makes it hard for other developers to do the release if the details weren't specified. This indicator, can only be evaluated by developers since they are the ones proceeding with the releases;

- **Success on dealing with conflicts**, with the use of CD, each feature development is encapsulated in a developing branch (common practice in CI). This branch is an image of production for the time development started. While the feature is developed new ones are releases, therefore, the previous branch will be outdated comparing to production. CI has the ability to easily resolve this issues,

In order to compile the results of the surveys quickly, they will be performed using google forms since it's a commonly used tool in the company and provides data analysis in the end. The questions to be asked are set out in Annex 1 (REFERENCIA A ANEXO).

5.1.4 Hypothesis Testing

In the future, after the completion of the current project, it will be possible to test if the pipeline has improved the release process by analyzing the results from the surveys as all as t-test will be used to confirm the improvement of KPI's.

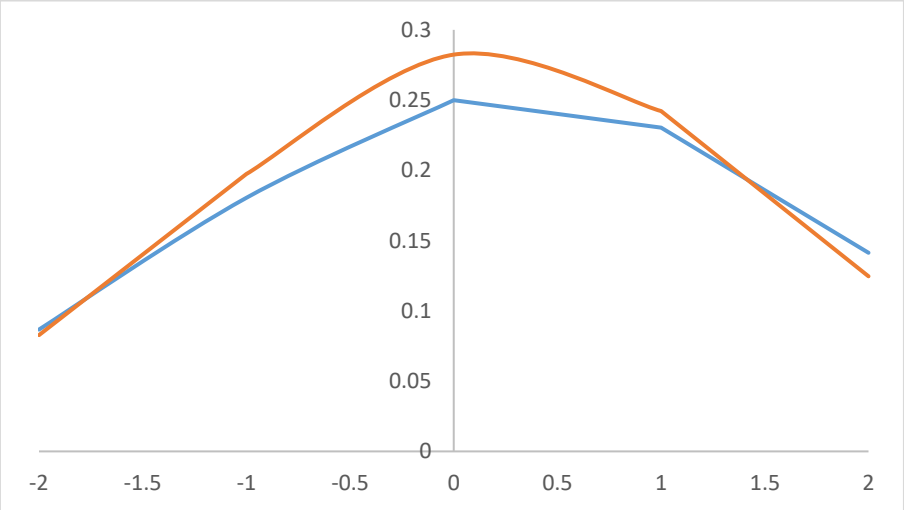
T-Tests are showed to be the best approach for testing this solution because each person will evaluate the "before" and "after" scenarios and rate each question from -2 to 2 for both of them. As there is one person two scenarios, this makes them paired (Fritz et al. 2015). This method allows the evaluation of any two scenarios, in this work two solutions for the same problem will be compared by the stakeholders and engineers of the project. Each question

will have an aggregate table of all answers collected, as it can be observed on Table 3, the set of numbers was generated randomly. Each table will be analyzed to reach a conclusion about the quality improvement comparing both solutions, featuring 10 pairs of observations.

Table 3 - Representation of answers collected for one question

Question 1		
	Current release process	**New release process**
User 1	0	1
User 2	2	-1
User 3	-1	0
User 4	-2	1
User 5	1	-1
User 6	2	-2
User 7	-2	2
User 8	0	1
User 9	2	-1
User 10	1	2

With this table it is possible to get the *t* distribution, which plays an important role in hypothesis testing. For this case, the distribution shows the average users opinion and how it is distributed, making it possible to test the development success.



Graph 1 – Example of normal distribution for one trial

6 Conclusion

This work addresses one BI projects gap, which is the lack of automatisms for releases. As a case study, Farfetch's BI projects were used.

CD concepts provide teams with a faster and easier method of releasing software, starting with the use of automatic tests and version control. But setting up a CD pipeline can be challenging depending on the team's current development methodology. For BI projects, version control tends to be non-existent so the first steps to use CD are for setting up version control and improve development methodologies.

The challenges for BI projects is finding the right tools and methods that fit each team necessities and requirements, because of the diversity of BI components it is difficult to find one universal tool, so manage each part of a BI project is an option, using one tool for database versioning, another one for ETL versioning and then, another for orchestrating the pipeline. At the same time, the organization aims at keeping software homogeneous through all platforms, so, it was necessary to find a balance between these.

The use of CD will improve software release frequency, since releases are manual and are liability, and software quality will improve because fewer bugs should be released into production thanks to the test stages before the final deploy to production.

Has it was previously referred, in the introduction chapter, the main goal is to reach WARP releases, teams are fully responsible for their release process and there is no external dependency with Tech-Ops. For reaching this stage of releases, teams need to be fully adapted to the current process of CD.

Implementing CD to BI projects has proven to be a challenge regarding version control of the ETL project because of its specific behavior in face of changes. The chosen solution, for this part is to completely replace the package with the new one, conflicts need to manually handle because of the lack of tools to merge changes. For the moment, no database versioning will be used. Because of the current law changes regarding data, new solutions are being researched, that both protect data and object changes. After choosing one of the proposed

solutions, the implementation was divided in stages to facilitate development and to successfully test each stage. This was proven successful for the pilot project which goal was to prove this hypothesis success.

Regarding future work, testing this new solution is the main goal, proving its success. Measuring the improvement in release frequency and the pipeline success is crucial. The architecture implemented should be visited and improved whenever necessary, further reading into existing tools is essential to make sure the solution evolves and with each breaking change, a new set of tests should be executed, proving the teams with the improvements through time.

References

Alex Papadimoulis, 2012. DATABASE CHANGES DONE RIGHT. Available at: http://thedailywtf.com/articles/Database-Changes-Done-Right.

Allee, V., 2002. Understanding Value Networks.

Ambler, S.W. & Sadalage, P.J., 2006. *Refactoring databases : evolutionary database design*, Addison Wesley.

Anon, 2017. About Flyway - Flyway by Boxfuse • Database Migrations Made Easy. Available at: https://flywaydb.org/about/ [Accessed February 15, 2017].

Anon, 2016a. ETL Tools Comparison | Best ETL Tools in market | Adeptia. Available at: https://adeptia.com/products/etl_vendor_comparison.html [Accessed February 15, 2017].

Anon, 2010. Flyway User Guide. Available at: https://flywaydb.org/documentation/command/migrate [Accessed February 15, 2017].

Anon, 2015. Talend MDM Platform Studio 6.0.1 - User Guide (EN) - Talend Online Documentation & Knowledge Base. Available at: https://help.talend.com/display/TalendMDMPlatformStudioUserGuide60EN/1.3+Teamwork+and+development+consolidation [Accessed February 15, 2017].

Anon, 2014. The Definitive Guide to Database Version Control. Available at: https://www.infoq.com/articles/Database-Version-Control [Accessed February 15, 2017].

Anon, 2016b. What Is Talend: Talend Software Overview & About Talend. Available at: https://www.talend.com/about-us/ [Accessed February 15, 2017].

Ballard, C., 2010a. IBM InfoSphere Streams. , (May). Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:IBM+InfoSphere+Streams#3.

Ballard, C., 2010b. IBM InfoSphere Streams. , (May).

Beck, K. et al., 2001. Principles behind the Agile Manifesto. Available at: http://agilemanifesto.org/principles.html [Accessed February 9, 2017].

Chacon, S., 2009. Pro Git. *Control*, pp.1–210. Available at: http://www.springerlink.com/index/10.1007/978-1-4302-1834-0$%5C$nhttp://books.google.com/books?hl=en%257B%257B%257D%257B&%257D%257B%257D%257Damp;lr=%257B%257B%257D%257B&%257D%257B%257D%257Damp;id=qJsXefpx1AUC%257B%257B%257D%257B&%257D%257B%257D%257Damp;oi=fnd%257B%257B%257D%257B&%257D%257B%257D%257Damp;pg=PR14%257B%257B%257.

Costa, J., 2015. BI - Continuous Deliver Using SSDT, SSIS and Git.

DANIEL PERIANEZ, 2014. Continuous integration in data warehouse development | CONCENTRA. Available at: https://www.concentra.co.uk/blog/continuous-integration-data-warehouse-development [Accessed February 9, 2017].

Dictionaries, O., 2017. Definition of value. Available at: https://en.oxforddictionaries.com/definition/value.

Dictionary, C., 2017. Perceived Value. Available at: http://dictionary.cambridge.org/dictionary/english/perceived-value.

Duvall, P., Matyas, S. & Glover, a, 2007. *Continuous integration: improving software quality and reducing risk*, Available at: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf%5Cnhttp://books.google.com/books?hl=en&lr=&id=PV9qfEdv9L0C&oi=fnd&pg=PP5&dq=Continuous+Integration.+Improving+Software+Quality+and+Reducing+Risk&ots=mThO_bQgyx&sig=LwhiiRZnlXNPkbX3dZHUB8lFsjI%5Cnh.

Factor, P., 2014. Continuous Delivery and the Database. Available at: https://www.simple-talk.com/sql/database-administration/continuous-delivery-and-the-database/ [Accessed February 13, 2017].

Fowler, M., 2013. Continuous Delivery. Available at: https://martinfowler.com/bliki/ContinuousDelivery.html [Accessed February 9, 2017].

Frank, J.H. & Zeng, L., 2013. On event-driven business integration. *Proceedings - 2013 IEEE 10th International Conference on e-Business Engineering, ICEBE 2013*, pp.82–89.

Fritchey, G. & Skelton, M., 2015. Database Lifecycle Management Achieving Continuous Delivery for Databases.

Fritz, M. et al., 2015. Chapter 3 – Comparing two designs (or anything else!) using paired sample T-tests. *Improving the User Experience Through Practical Data Analytics*, (1993), pp.71–89.

Harris, J., 2016. The growing importance of big data quality. Available at: http://blogs.sas.com/content/datamanagement/2016/11/21/growing-import-big-data-quality/.

Herstatt, C. & Von Hippel, E., 1992. Developing New Product Concepts Via the Lead User Method: A Case Study in a “Low Tech” Field", Lead User Workshops for New Product Concept Development: A Case Study. *Journal of Product Innovation Management*, 9, pp.213–221.

Humble, J., Farley, D. & Fowler, M., 2010. *Continuous Delivery Reliable Software Releases Through Build, Test and Deployment Automation*,

Hurwitz, J. et al., 2013. *Big Data for Dummies*,

IBM, 2016. IBM - InfoSphere Information Server - Data Integration, Information Integration - Overview. Available at: https://www-01.ibm.com/software/data/integration/info_server/ [Accessed February 14, 2017].

Khorikov, V., 2015. State vs migration-driven database delivery - Enterprise Craftsmanship. Available at: http://enterprisecraftsmanship.com/2015/08/18/state-vs-migration-driven-database-delivery/ [Accessed February 13, 2017].

Kimball, R., *The Data Warehouse Toolkit. The Complete Guide to Dimensional Modelling-Third Edition*,

Kimball, R. & Caserta, J., 2015. *The Data Warehouse ETL Toolkit*,

Kphsuke Kawaguchi, 2017. Jenkins Wiki. Available at: https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins.

Luhn, H.P., 1958. A Business Intelligence System. *IBM Journal of Research and Development*, 2(4), pp.314–319. Available at: http://altaplana.com/ibmrd0204H.pdf.

Methods, E. et al., 1996. Fuzzy Front End : and Techniques. *Industrial Research*.

Minium, D., 2006. Team Foundation Server Fundamentals: A Look at the Capabilities and Architecture. Available at: https://msdn.microsoft.com/en-us/library/ms364062(v=vs.80).aspx.

NICOLA, S., FERREIRA, E.P. & FERREIRA, J.J.P., 2012. A NOVEL FRAMEWORK FOR MODELING VALUE FOR THE CUSTOMER, AN ESSAY ON NEGOTIATION. *International Journal of Information Technology & Decision Making*, 11(3), pp.661–703. Available at: http://www.worldscientific.com/doi/abs/10.1142/S0219622012500162 [Accessed February 9, 2017].

Noone, S., Agile Mindset. Available at: https://www.linkedin.com/pulse/agile-mindset-simon-noone.

Octopus, 2017. Octopus Documentation. Available at: https://octopus.com/docs/getting-started#Gettingstarted-Octopusinyourdeliveryprocess.

Osterwalder, A. et al., 2010. *Business Model Generation*, Available at: http://www.amazon.com/Business-Model-Generation-Visionaries-Challengers/dp/0470876417.

Porter, M.E., 1985. Competitive Advantage - Creating and Sustaining Superior Performance. *New York: FreePress*, p.580.

Stricker, N. et al., 2017. Considering Interdependencies of KPIs – Possible Resource Efficiency and Effectiveness Improvements. , 8(October 2016), pp.300–307.

Swoyer, S., 2016. DevOps and BI: Software Development in Transition -- Upside. Available at: https://upside.tdwi.org/articles/2016/02/24/devops-and-bi.aspx [Accessed February 10, 2017].

Weber, A. & Thomas, R., 2005. Key Performance Indicators - Measuring and Managing the Maintenance. *IAVARA Work Smart*, (November), pp.1–16.

Windows, M. et al., 2014. *Data Warehousing in the Age of Big Data*, Available at:

http://www.americanbanker.com/issues/179_124/which-city-is-the-next-big-fintech-hub-new-york-stakes-its-claim-1068345-1.html%5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/15003161%5Cnhttp://cid.oxfordjournals.org/lookup/doi/10.1093/cid/cir991%5Cnhttp://www.scielo.

Woodall, T., 2003. Conceptualising “Value for the Customer”: An Attributional, Structural and Dispositional Analysis. *Academy of Marketing Science Review*, 12(5), pp.1–42.

Yaniv Yehuda, 2014. Database Continuous Delivery. Available at: https://www.infoq.com/articles/Database-Continuous-Delivery.

Yates, A., 2015. Critiquing two different approaches to delivering databases: Migrations vs state | working with devs... Available at: http://workingwithdevs.com/delivering-databases-migrations-vs-state/ [Accessed February 13, 2017].

Attachments

Annex 1

FARFETCH

September 2016

Name (optional):
Department:

Business Intelligence feedback survey

The BI team started using a release pipeline, continuing Farfetch's use of Continuous Delivery thought out all tech. This aims at improving the development and release process, having has mains beneficiaries you (stakeholders and engineers). We want to know your opinion to understand the impact of the pipeline and if you undergone any improvement after the use of Continuous Delivery.

Please grade the following points according to your experience, where 0 means no improvement and 5 means improved significantly.

Indicator	0	1	2	3	4	5
Time to request fulfillment						
Software performance						
Number of bugs detected						
Downtime experienced during and after release						

Only qualify the following indicators if you are an engineer for the data cluster.

Indicator	0	1	2	3	4	5
Release time						
Ease of use						
Success on dealing with conflicts						