# Detection of Delay Points in the Batch Release of Products in a Pharmaceutical Company

**André Miguel Machado Lopes**

Thesis to obtain the Master of Science Degree in

## Industrial Engineering and Management

Supervisor: Prof. João Pedro Bettencourt de Melo Mendes

### Examination Committee

Chairperson: Prof. Ana Isabel Cerqueira de Sousa Gouveia Carvalho
Supervisor: Prof. João Pedro Bettencourt de Melo Mendes
Member of the Committee: Prof. Maria Isabel Craveiro Pedro

**January 2019**

# Acknowledgements

I would first like to thank my thesis supervisor Professor João Pedro Mendes. The door to Prof. Pedro Mendes' office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work but steered me in the right direction whenever he thought I needed it.

I would also like to thank the Company personnel who were involved in this project: Country Quality Assurance Lead Portugal, Quality Assurance Manager and Head Supply Chain Manager. Without their passionate participation and input, the project could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents and to my girlfriend Carolina for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

André Lopes

# Abstract

The Company is a multinational enterprise that produces and sells pharmaceutical products in several European and external markets. In Portugal, the Company only oversees the distribution process, hiring a Pre-Wholesaler, with agreed conditions and standards, to receive, store, and verify the transport conditions of products. Upon registering each arrival into the information system, the Pre-Wholesaler sends samples to the quality control department of the Company. Products that meet standards are released to a Wholesaler (which then distributes them to the retail). This process is called Batch Release. Over the years, the process has become more complex due to increasingly stringent European and national pharmaceutical regulation and to the service level the Company wants to offer its clients. Company officers say that what once took one day, now takes nine. Any reduction in the process duration is likely to carry a significant reduction in costs tied up as inventory. The process was first described in Systems Modelling Language (SysML) to detect delay points in the system. Then, this model was used as a reference to specify the foundation of a Blockchain technology application, which makes it possible to know the reasoning behind the delays and cope with them.

A Empresa é uma multinacional que produz e vende produtos farmacêuticos em mercados europeus e externos. Em Portugal, a Empresa apenas supervisiona o processo de distribuição, contractando um Pre-Grossista qualificado para receber, armazenar e verificar as condições de transporte dos produtos. Após registar a chegada dos lotes, o Pre-Grossista envia amostras dos produtos para o departamento de controlo de qualidade da Empresa. Caso os produtos estejam conforme, estes podem ser aviados para o Grossista que os distribui pelas lojas. Este processo, que tem o nome de Libertação, ao longo dos anos complicou-se. Isto deve-se à regulamentação europeia e nacional sobre produtos farmacêuticos e ao nível de serviço que a Empresa pretende oferecer aos seus clientes. O relato dos gestores na Empresa diz que o que inicialmente demorava 1 dia, agora tem a duração de 9. Qualquer redução na duração do processo é provável que tenha um grande impacto na redução de custos de inventário. Utilizando o SysML (Systems Modelling Language) como ferramenta de modelação, o sistema foi descrito, detetando os pontos de atraso neste processo. Esta descrição foi usada como uma estrutura de referência para criar a base de uma aplicação da tecnologia Blockchain, que torna possível saber a razão pelo qual existem os atrasos e como os mitigar.



# Keywords

Pharmaceutical Industry, Supply Chain, Batch Release, Delay, SysML, Blockchain

# Table of Contents

# Index of Figures

# Acronyms

act – Activity Diagram

AQWA – Adaptable Quality Workflow Application

bdd – Block Definition Diagram

BRC – Batch Release Certificate

CoA – Certificate of Analysis

CCIS – Cold Chain Information System

Dragon – Drug Regulatory Affairs Global Compliance

DoC – Duty of Care

DoDAF – United States Department of Defense Architecture Framework

EU – European Union

EUBC – European Union Batch Certificate

EUBRC – European Union Batch Release Certificate

ibd – Internal Block Diagram

KPI – Key Performance Indicator

MBSE – Model-Based Systems Engineering

par – Parametric Diagram

pkg – Package Diagram

QA – Quality Assurance

QP – Qualified Person

QPEU – European Union Qualified Person

QPPT – Portuguese Qualified Person

SAP – Systems, Applications and Products

sd – Sequence Diagram

Software A – CCIS

Software B – Dragon

Software C – AQWA

Software D - SAP

stm – State Machine Diagram

SysML – Systems Modelling Language

uc – Use Case Diagram

UML – Universal Modelling Language

**Obs**: The Company uses other acronyms besides these, described in a sixty-five page manual.

# 1. Introduction

## 1.1 Background

Quality is of the utmost importance in the Pharmaceutical Industry due to heavy national and European regulation. Meaning, failure to comply with quality regulation has a substantial consequence for the company. For this reason, the pharmaceutical industry's supply chain is long and complex, with several quality verification steps, starting during production, up until after the product's arrival at the retailer. These checks are required in order to prevent the release of faulty products to the market. Figure 1 shows a simplified diagram of the Pharmaceutical Industry's supply chain. The Batch Release is an important step in this supply chain where, after arriving at the pre-wholesaler's warehouse, the products are verified for quality and cleared for distribution to wholesalers.

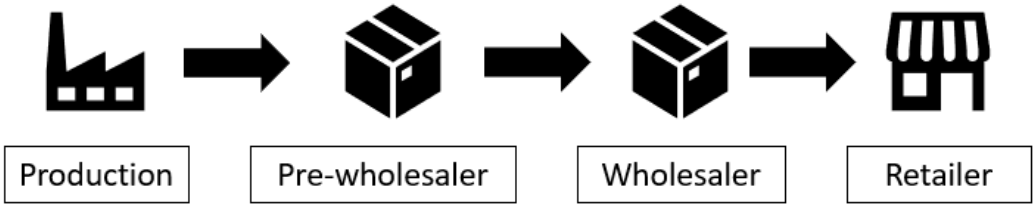


*Figure 1 - A simple illustration of the Pharmaceutical Industry's supply chain*

## 1.2 Problem Definition

The Company's supply chain has associated costs, one of which is the inventory cost from holding products in stock. The total inventory cost is proportionate to the time the stock stays on hold in the warehouse, meaning the longer the products stay stored, the higher the cost. The Batch Release stage currently has a duration of nine days, while it took one in the past, and the inventory on hold has increased in proportion. Sized to 5M€ waiting to be released, it is a concern for the Company.

## 1.3 Project Objectives

The main goal of this project is to detect the delays in the Company's batch release process and providing a way to find the reasoning for these delays, in order to cope with them afterwards.

This project uses the Model-Based Systems Engineering (MBSE) methodology to identify the problem causing these symptoms, the delays. MBSE is the formal application of modeling to support system design, analysis, optimization, verification and validation. It begins in the conceptual phase, and may continue throughout development and into subsequent system life cycle phases such as operations [1]. This application of MBSE uses Systems Modelling Language (SysML) as the modeling language to formally describe and analyze the system. SysML is a general-purpose modelling language for complex engineering systems [2]. It uses a graphical notation to support the analysis, design and verification of systems, including facilities, procedures and personnel [3].

The first phase of this project was the development of a reference structure for the Batch Release Process. This phase had the purpose of identifying how much time is spent in each stage of the process and what are the delay points within the system.

The second phase used the reference structure to develop the foundation of a Blockchain technology procedure, as a means to cope with the issues found. The Blockchain results will help evaluate the performance of each step (e.g. the Pre-Wholesaler verification) and the need for redundant steps.

## 1.4 Dissertation Structure

This thesis is structured as illustrated in Figure 2. The first section, State of the Art, provides an overview explanation about the MBSE and Blockchain concepts to better understand how to apply them to the project. It includes the main features, a bit of the history and a few application examples, as well as detailing the tools which apply the MBSE and Blockchain concepts, those being the SysML and IBM Blockchain Platform Software, respectively.

The first section builds up to the second one, Methodology, which is a description of the project methodology, detailing how the concepts of the previous section were applied to the project.

Finally, the second segment leads to the third and final section, Results and Discussion, which shows the resulting SysML model and Blockchain model file, and its validation through the creation of a sample population. Also included are a discussion and details about possible future work.

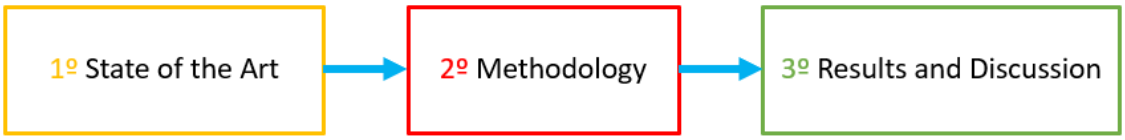


*Figure 2 - The Dissertation Structure*

# 2. State of the Art

This section details MBSE's key futures, history, some different approaches to system design and the tool used to apply this method, SysML. It also overviews the Blockchain main concepts, some of its practical applications currently in use and the IBM Blockchain Platform, a software used to create applications with this technology.

## 2.1 MBSE

### 2.1.1 Key Futures

Model Based Systems Engineering (MBSE) is a methodology that applies systems modelling to support analysis [4], design [5] [6] [7] [8] [9], specification [10] [11] and validation [4] [12] of a system being developed [13] [14]. MBSE uses Systems Modelling Language (SysML) as a tool in order to create a cohesive model of the system from specification, improving design quality and communications among the development team [7] [12] [13].

SysML is a visual modeling language that was adapted from the Unified Modelling Language (UML) to enhance the traditional top-down systems engineering process [13] [14] [15]. Although the UML is pretty extensive, SysML added some needed elements for systems engineers while removing others, focused on information systems modelling, to diminish the complexity of the tool [14].

The SysML uses a simple diagram approach to describe systems, where basic structure components, called Blocks, are used to represent the elements of the system [2] [13] [14] [15]. In Figure 3, the structural view represents how the different elements that define the structure of the system, the Blocks, are combined. The behavioral view describes the activities, states and users in which the blocks from the structural model are involved with [14] [15].

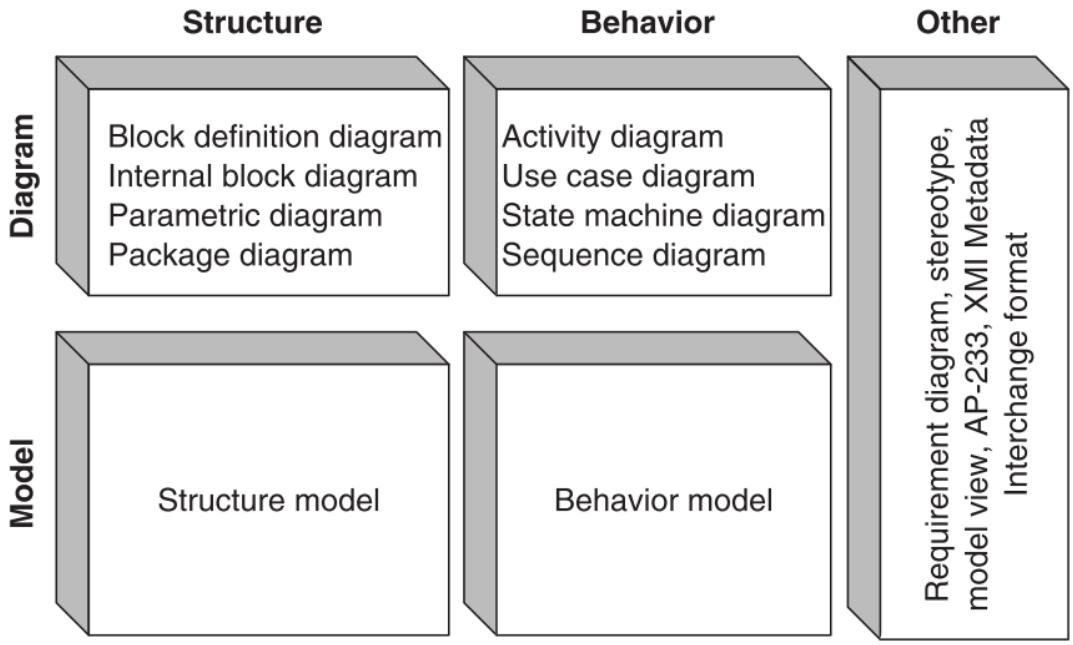


*Figure 3 - SysML model types and associated diagrams* [14]

### 2.1.2 Systems Engineering History

The first use of the term *systems engineering* dates back to the 1940s in Bell Telephone Laboratories and describes its major applications during World War II [15]. The National Defense Research Committee in conjunction with Bell Laboratories, created a Systems Committee to support a project called C-79, which its first task was to improve the communication system of the Air Warning Service [15]. In 1951, systems engineering was recognized as a unique function in the Bell Laboratories organizational structure [15].

One of the most well-known early application of systems engineering was its involvement with the earliest Intercontinental Ballistic Missile (ICBM) program, known as Atlas, in 1953 [15][16].

Due to the growing complexity of commercial fields, the need for integrated methods rose. In 1962, Arthur Hall published “Methodology for Systems Engineering”, which came from the telecommunication industry [14].

In 1990, an organization named National Council on Systems Engineering (NCOSE) was founded in the United States, and its main goal was the promotion and development of systems engineering in the country. After 5 years, its goal was extended to the international level, being renamed International Council on Systems Engineering (INCOSE) [14].

### 2.1.3 Systems Engineering Approaches

Since the 1950s, modelling techniques are used for system design. These methods describe system components, dynamic system behavior and detail system functions.

One of the specification languages, the Unified Modelling Language (UML), was created by object-orientated programmers, who use a bottom-up design process approach to describe systems. This begins by defining a group of objects which are part of a system in order for it to achieve its desired functionality. The UML’s components are obtained from more specific system objects that were coded or adapted from existing code. This means most of the model design is the code itself, so there is not a lot of modeling or analysis using this approach [13][15].

The United States Department of Defense Architecture Framework (DoDAF) provides three top-level views needed to describe a system. These are the operational, systems and technical views. Each of these is composed of subviews that use textual, graphical and tabular descriptions. A data model is created to define the entities and relationships between the data elements which make these integrated views. Although the DoDAF has a structure that can work for all types of systems, it is too complex and cumbersome for specific systems [13][15].

Due to the need of design analysts to find errors and stress points in the system, systems engineers created model-based approaches, the most known of which is the Systems Modelling Language (SysML). Besides being easier to implement, this language was adapted from the UML to have much greater emphasis on system behavior instead of interaction, meaning it’s a lot more efficient to provide information for design reviews. One SysML main challenges is for the system description to be easily understood by non-engineering stakeholders [13][15].

## 2.2 SysML Overview

As stated, SysML contains two types of views, structural and behavioral, containing four kinds of diagrams each. This segment describes the diagrams as well as some important elements frequently used together with them.

### 2.2.1 Block Definition Diagram

The Block Definition Diagram (**bdd**) states the structural elements of the system, the blocks, and their classification and composition.

In Figure 4, the block “Card Reader” is defined with its respective attributes and parts.

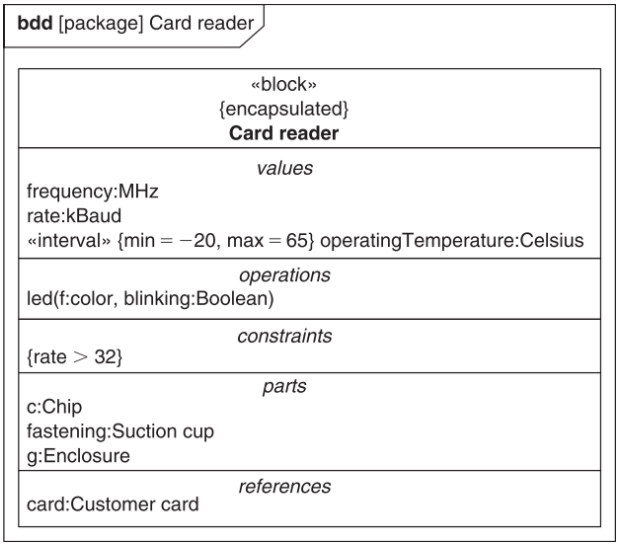


*Figure 4 - An example of a Block Definition Diagram* [14]

### 2.2.2 Internal Block Diagram

The Internal Block Diagram (**ibd**) shows how the parts of each block are interconnected and interact with each other.

In Figure 5, the interaction between the parts of a Pump system block is illustrated. The part Pump, transfers water, between two containers, which are also parts of the pump system.

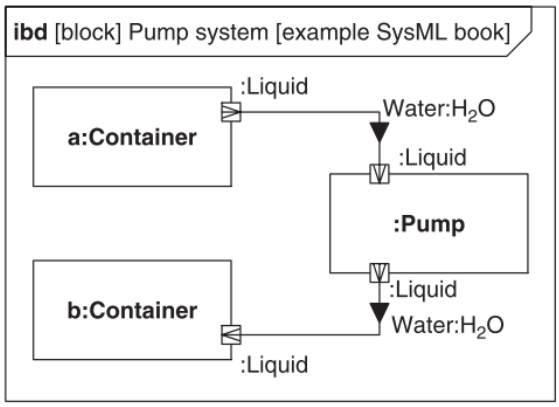


*Figure 5 - An example of an Internal Block Diagram* [14]

### 2.2.3 Parametric Diagram

A Parametric Diagram (**par**) supports engineering analysis by defining relations, like $E = m.c^2$, called Constraints, between properties or parameters of different Blocks.

In Figure 6's example, the Mass Relationship (d*V = m) is represented by a constraint which relates the parameters volume, density and mass.

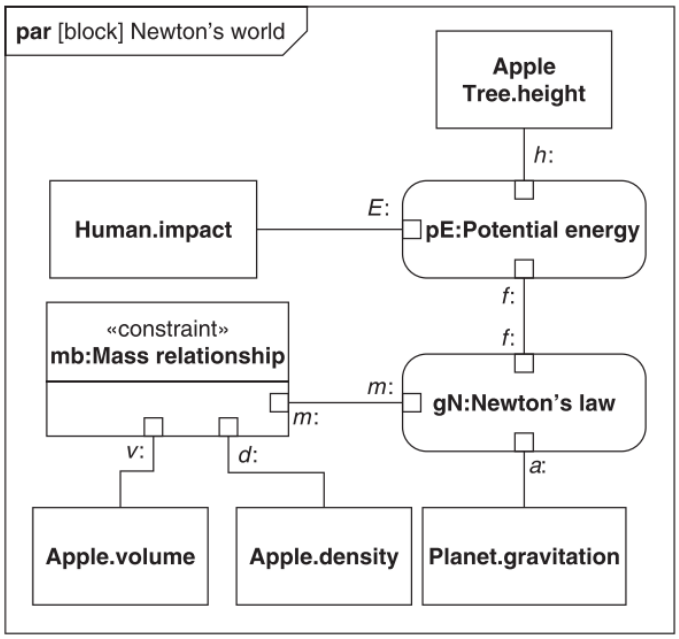


*Figure 6 - An example of a Parametric Diagram* [14]

### 2.2.4 Package Diagram

Package Diagrams (**pkg**) organize and structure model elements in packages, similar to how folders group files by category. Figure 7 shows an example of a Package Diagram.

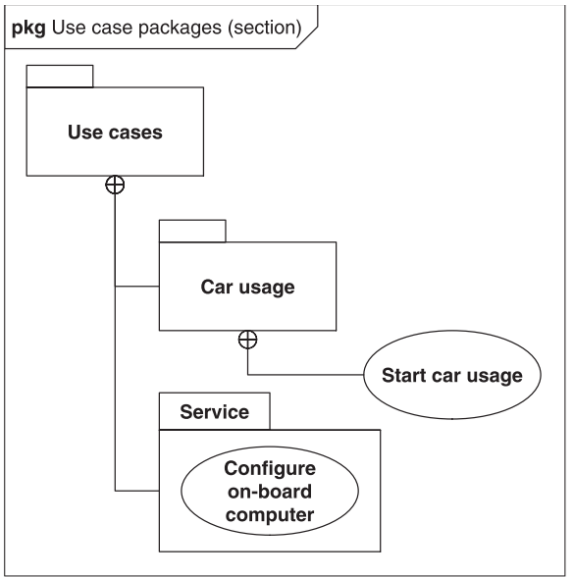


*Figure 7 - An example of a Package Diagram* [14]

### 2.2.5 Activity Diagram

The Activity Diagram (**act**) presents flow-based behaviors, indicating in which order are actions executed which depend on the availability of inputs, outputs and outside control, and how these actions transform inputs into outputs.

In Figure 8, the order of each of the actions is displayed, as well as where the flow starts and what entities of the system realize these actions.

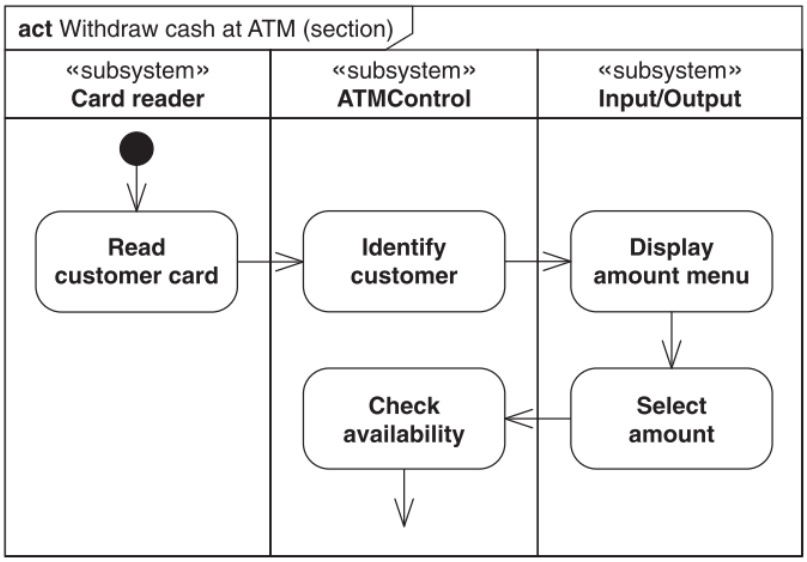


*Figure 8 - An example of an Activity Diagram* [14]

### 2.2.6 Use Case Diagram

The Use Case Diagram (**uc**) describes the services a system offers and as well as the external entities that use them, named Actors, to achieve a certain goal.

In the example displayed in Figure 9, the Actor *Customer* performs the action *Buy Soft Drink*, which is a feature the system has.

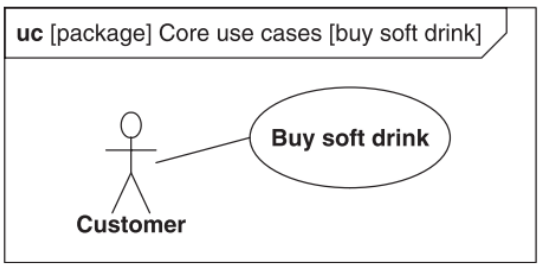


*Figure 9 - An example of an Use Case Diagram* [13]

### 2.2.7 State Machine Diagram

A State Machine Diagram (**stm**) presents how an entity behaves based on its transitions between states, which are triggered by events.

In Figure 10, a State Machine Diagram is represented. Here is described how a railroad crossing changes state. The crossing is in the state *open* until a train arrives, then it changes to *block* until the

the railroad crossing gate reaches the down position, where it changes state to *closed.* It's important to note that some states have an event triggered by entering, exiting or being in a certain state. In the example, when the railroad crossing enters the *block* state, a blinking light will turn on. This light only stops when exiting the state *release.*

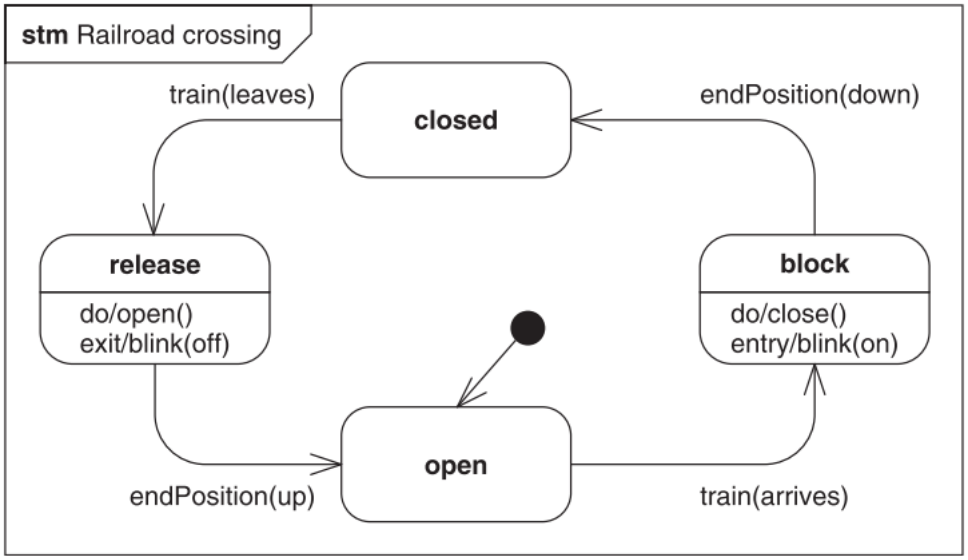


*Figure 10 - An example of a State Machine Diagram* [14]

State Machine Diagrams are similar to Grafcet, which describe a machine behavior step-by-step. As such, Grafcet has its basic elements in common with this type of diagram: steps, transitions and statements that describe the actions in each step and the trigger of each transition. An example of a Grafcet is provided in Figure 11.

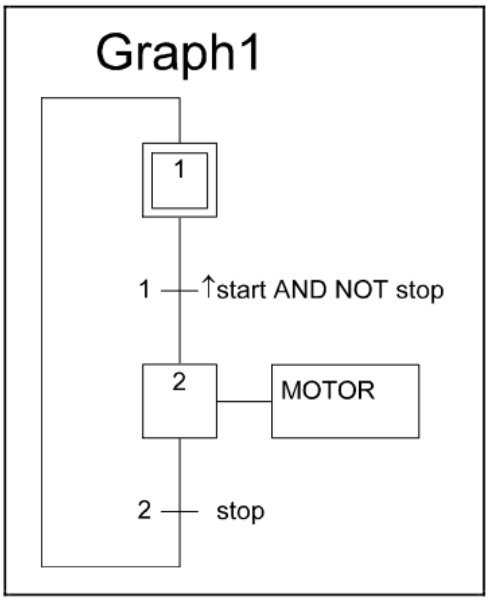


*Figure 11 - An example of a Grafcet* [17]

### 2.2.8 Sequence Diagram

The Sequence Diagram (**sd**) shows the behavior of an entity in terms of the sequence of messages traded between systems or elements of systems.

In the example presented in Figure 12, two entities of the system *Turn On Vehicle* are shown interacting with each other through a sequence of messages. The sequence is read chronologically from top to bottom, meaning *driver* sends a message to *vehicle* saying *ignition on*. Upon receiving this message, *vehicle* responds to *driver* with a *vehicle on* reply. Messages can be triggered by events that happen within the system. The time between messages can also be represented on a Sequence Diagram by a double ended vertical arrow with the respective time period associated.

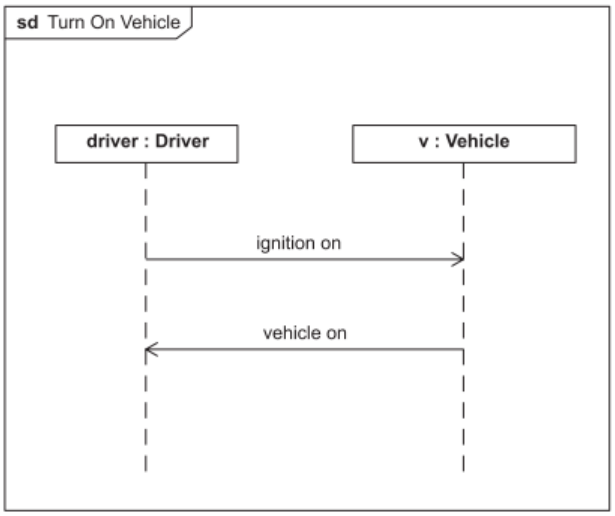


*Figure 12 - An example of a Sequence Diagram* [13]

Figure 13 shows a sequence diagram where the time period between the messages *test cameras()* and *System OK* is shown.

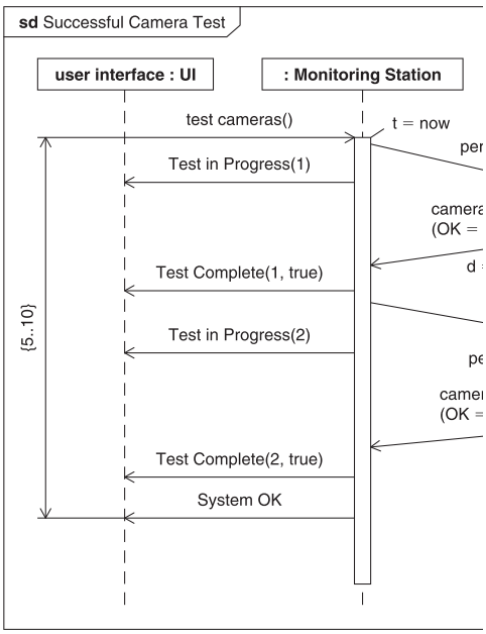


*Figure 13 - Sequence Diagram example* [13]

### 2.2.9 Generalization

The term Generalization refers to taxonomic relationship from a specialized element to a general one. This is represented in the diagrams by a hollow arrow going from the specialized entities to the general ones. Generalization also means, the specialized element inherits all of the properties of the general element, but not the other way around.

In the example given in Figure 14, the blocks *Wired Camera* and *Wireless Camera*, inherit all the parts from the block *Camera*, since the latter is a generalization of the first two.

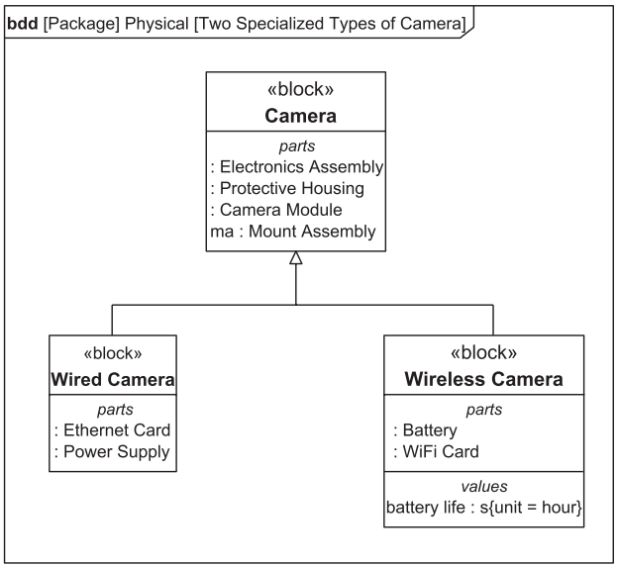


*Figure 14 - An example of a block generalization* [13]

### 2.2.10 Association

An association represents the relation between two blocks. These can have multiplicity, meaning, more than one of the same block can be related to one or more of another block.

In the example presented in Figure 15, the *Command Center* and *Surveillance System* blocks are associated by an *ADSL Connection*. It can also be noticed that *Command Center* can be connected with any number of *Surveillance System*, but *Surveillance System* must be connected to one and only one *Command Center*.

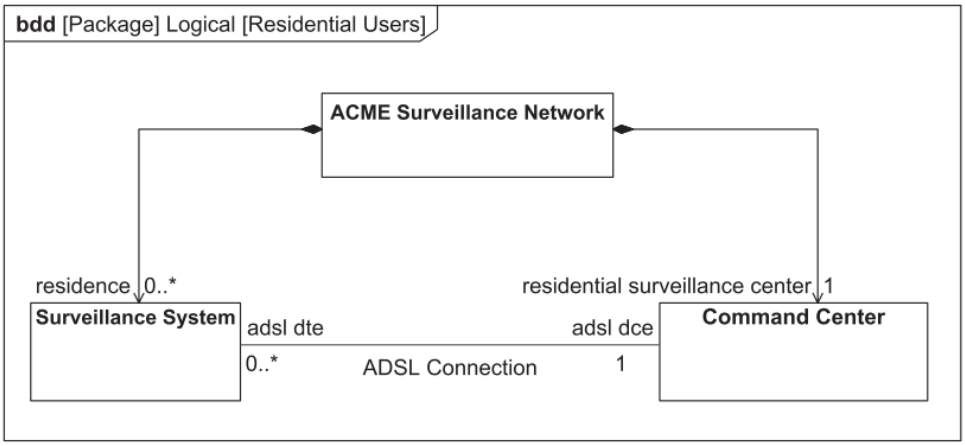


*Figure 15 - Association in a bdd, example* [13].

## 2.3 Blockchain

A Blockchain is a ledger of transactions that can be checked by any participant involved but is not subject to any form of central control [18]. It can have several applications, where its most famous one is the virtual currency *Bitcoin*, where it tracks transactions, facilitates money transfer and prevents double-spending, not requiring banks. Other uses include networks for supply chain [19], insurance [20], and content creation such as music [21]. On the other hand, Blockchain applications require a lot of server space, meaning, its implementation may be resource heavy. Although, the server space price is not as high as it once was, this being one of the reasons Blockchain is becoming fairly used nowadays [22].

### 2.3.1 Key Futures

Blockchain technology is a peer-to-peer network of information technology that keeps a record of digital transactions in a distributed ledger, accessible by any participants in the system [18]. A peer-to-peer is a type of network, where its maintenance tasks and responsibilities are shared between several devices, called peers [23]. This is opposed to the traditional databases, which are sited in centralized servers and controlled by a single mediator like a bank or the government. Being decentralized and cryptographic, blockchain technology builds trust between peers, therefore bringing major security benefits [24]. Hacker attacks that usually target the large mediators, such as banks, are unfeasible and virtually impossible, as the blockchain registers all alterations that would occur with the hacking attempt.

Figure 16 demonstrates how blockchain technology works through a simple example, an ice cream sale.

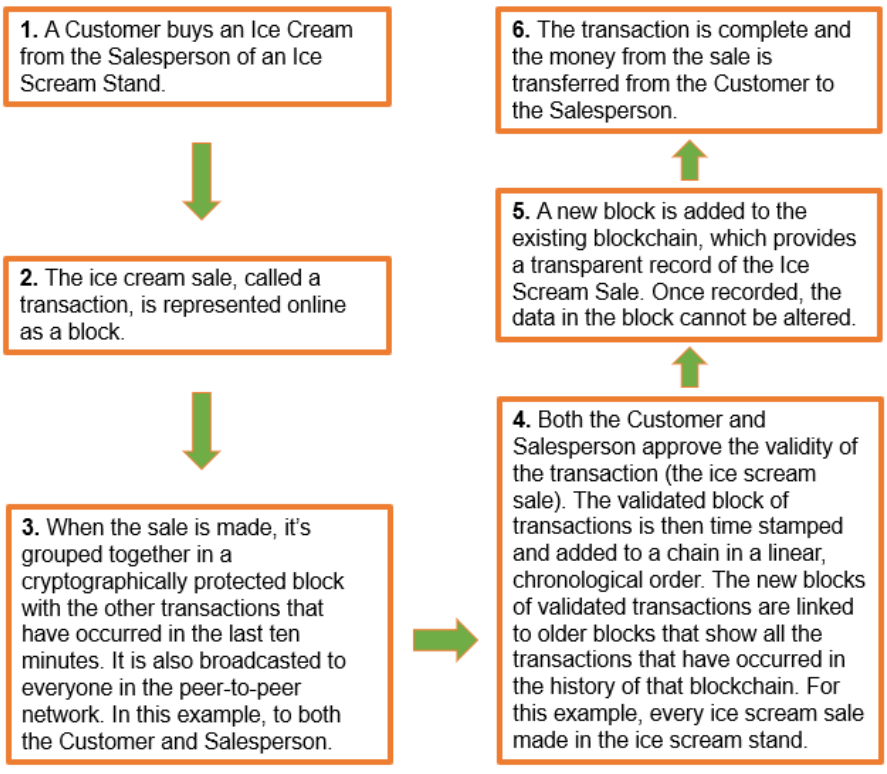


*Figure 16 - A demonstration of how blockchain works through an Ice Scream Sale example* [24]

As evidenced in the previous example, Blockchain solutions are based on four key futures, which make this technology very desirable to monitor large amounts of information while being secure at the same time.

- **Decentralized Validation** - All relevant peers in the network need to validate each transaction, otherwise it will not go through. For example, the transaction for a warehouse door opening for an incoming delivery truck will only occur if both the truck driver and the warehouse doorman participants approve it.
- **Redundancy** - The Blockchain is decentralized, meaning its replicated on all peers of the network. This results in the Blockchain being unaffected even if some of its storage hardware is lost. Figure 17 shows the comparison between two types of network. On the left, a decentralized network, where all the data and decisions are shared between some or all the devices in the network, in comparison with a centralized network on the right, where the information is centered in a single device.

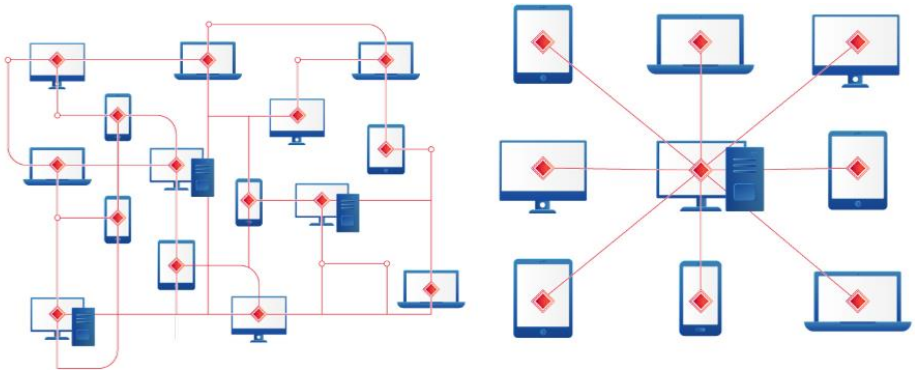

*Figure 17 - Comparison between a decentralized and a centralized network* [23]

- **Immutable Storage** - All blocks are linked, so tampering with one block requires changing all successive blocks. All registered data also has a digital fingerprint with timestamp. So, even if tampering is successful, the change stands out as obvious, as the new fingerprint does not correspond to the old one.
- **Encryption** - Due to transaction validation being made with digital signatures, which are based on pairs of cryptographic public and private keys, only the participants in the network can register data in the Blockchain. Figure 18 provides an overview on how public-key cryptography works.

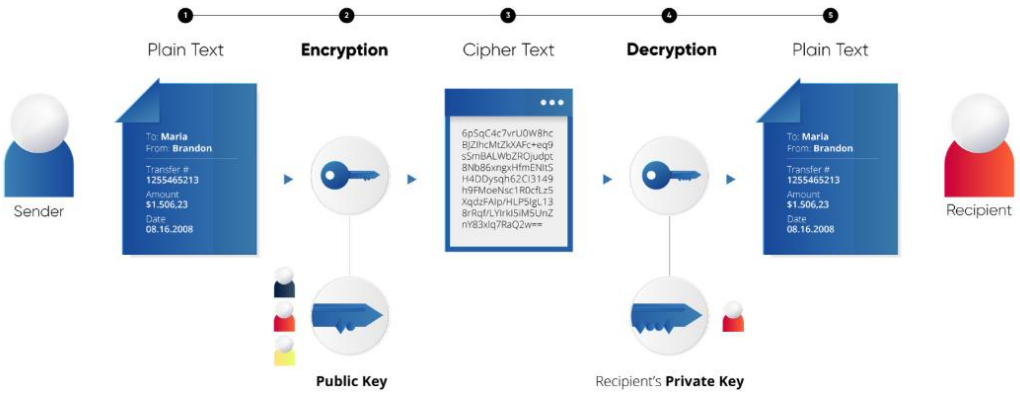


*Figure 18 - How public-key cryptography works* [23]

### 2.3.2 Blockchain Applications

Despite being a relatively recent technology, with its first use in 2008 with the creation of the virtual currency *Bitcoin* by an anonymous person or people going by the alias of Satoshi Nakamoto [25], Blockchain already has multiple applications in several business areas.

In supply chain, Blockchain is used by companies to monitor their products and parts no matter how numerous they are, or to check if the source of a product is adequate. Everledger Ltd is a company that uses Blockchain to help other corporations track the origin of diamonds, allowing buyers to check if the stones are mined in zones with forced labor, or if the results of previous sales were used to fund violence [19][26].

In the music industry, Blockchain is useful to create smart contracts between the parties involved, preventing piracy and crediting the artists. This is accomplished by associating the music files to a Blockchain, where the songs will only play after the required payment is made [7][8]. Ujo is a Blockchain platform created with this intent, where some musicians have already published their albums, like singer-songwriter Imogen Heap [8][9].

In the healthcare department, Blockchain can be used to monitor clinical activities and to store an accurate patient medical history. This is very critical for long-term treatment cases like cancer patients, where it's required to know former lab results, what medication and dose size the patient took and what are the possible negative side effects associated with the medicine [28]. With this intent, some private medical clinics in the Russian Federation have begun to apply this technology in their system [29]. The Blockchain network stores all patient data in a single electronic map, allowing for quick adaptation to the constant changes. This system can also be used regardless of the location of the clinics, as all data is stored in a single widespread database, enabling it to be processed in real time. It also allows for the collection of patient state of health information after being discharged [29].

Some learning institutions, such as the Sechenov University in Moscow, are using Blockchain in the training of medical workers, with the purpose of monitoring the skills, controlling the knowledge and verifying the level of education students have during the course [29].

In Estonia, its innovative digital identity, e-Residency, benefits greatly from Blockchain. E-Residency is a commercial initiative from the Estonian government, enabling the e-Residents to undertake commercial activities with the public and private sector of the country [30]. Although it's not a travel or a citizenship document, it can be considered an international passport for the digital world. Adding Blockchain technology to e-Residency would imply users could have unprecedented control over its identity information, while guaranteeing its authenticity [30].

## 2.4 IBM Blockchain Platform

The tool used to create a Blockchain business architecture for the Company is the IBM Blockchain Platform, which is an enterprise-ready development platform powered by the Hyperledger Composer open source engine [31]. Hyperledger Composer provides a Playground environment where the user can create and test their Blockchain business networks.

The Blockchain business architecture contains 3 elements:

- **Assets** represent the goods, services or property in the business. These can include anything from cars, animals, documents or workers. Assets can have many properties, requiring only a unique identifier. Some of these properties can be related to other assets in the network.
- **Participants** represent the business members who can own assets or submit transactions. Like assets, participants just require a unique identifier and may have many other properties.
- **Transactions** represent the interactions between participants and assets. An example is a sale. A participant sells an asset to another participant, therefore changing its ownership.

Figure 19 shows a business architecture, which is stored in an archive (.bna) composed of four types of files:

- **Model** file (.cto), usually created by the business analyst, describes the business environment by defining the structure and relationship between the three model elements: assets, participants and transactions.
- **Script** files (.js), usually created in JavaScript by developers implementing the system requirements provided by the business analyst.
- **Access Control** files (.acl) define the access rights of each participant in the business architecture, meaning each entity can only alter the data it has access to.
- **Query** File (.qry), states the details of the business architecture (e.g. name, version, etc.).

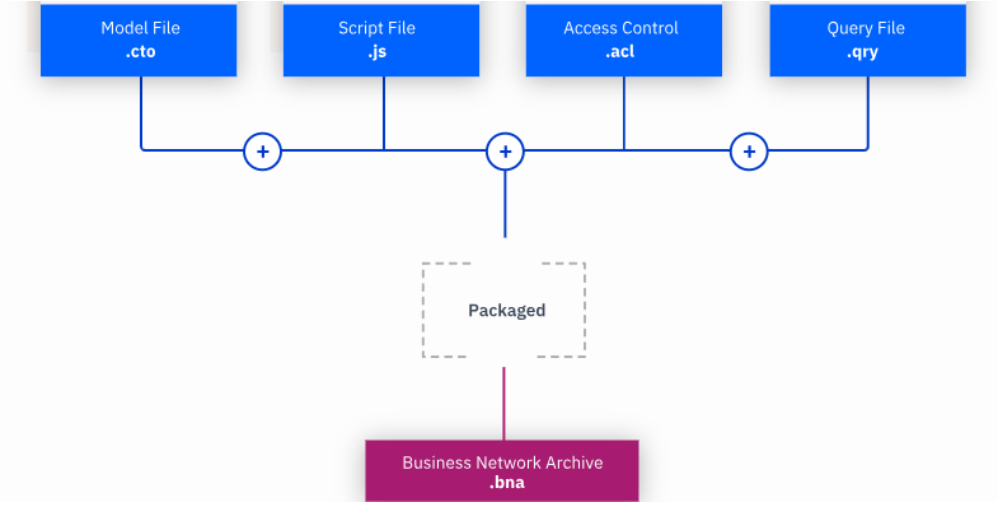


*Figure 19 - The elements that make a Blockchain Business Architecture Archive* [31]

A business architecture built in the Playground cannot interact with other software, but its .bna archive can be extracted and imported to IBM Blockchain Platform which can later be linked with external systems.

# 3. Methodology

This section overviews the two-step methodology used in this project:

- The first step was building a SysML model (.rpy) to illustrate the batch release process system with the information and data provided by the Company;
- In the second step, these diagrams were then used as a frame architecture for a Blockchain technology application.

Figure 20 illustrates this methodology. The Company information and data were converted into a SysML model (.rpy), which was later adapted to a Blockchain model file (.cto).

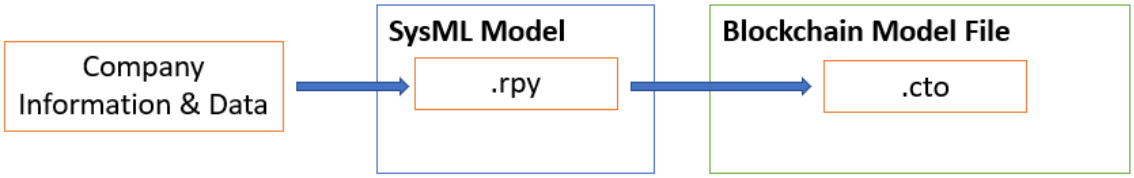


*Figure 20 - The two-step methodology used in this project.*

## 3.1 Project Diagram Sequence

The SysML model began with the building of a block definition diagram of the main structural elements of the system (**1**). This was followed by the creation of a use case diagram (uc) defining the main actors involved with the process and what actions they perform to interact with the system (**2**).

Two activity diagrams (act) were created from the uc's, one for the central batch release process and another for the local batch release process, which describe the sequence of activities in each of these processes (**3**). From the act's, a state machine diagram (stm) was built for each of the core blocks in the system, the Batch and the Duty of Care (DoC). These diagrams describe the lifecycle of the blocks, showing what states they go through and what are the triggering conditions for changing states (**4**).

Afterwards, a block definition diagram with variables was created (**5**). Variables offer a way to evaluate the performance of the system with provided data. Finally, using the variables and elements from the other diagrams, a sequence diagram (sd) for each of the core blocks of the system was created. These sd's allow checking the sequence of actions of the process, and how these actions affect the system and in what way they affect it (**6**).

This stage of the methodology is shown in Figure 21.

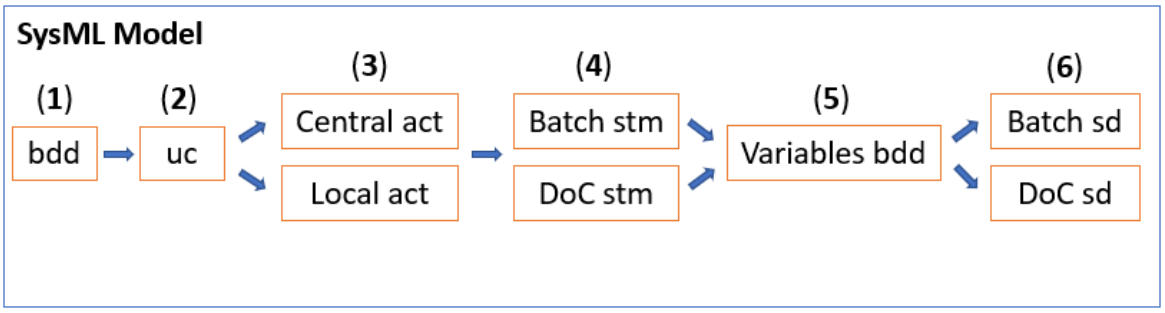


*Figure 21 - The diagrams used in the project and the sequence they were created.*

## 3.2 Blockchain Transition

Pairing elements from both software models showed several resemblances. This made the transition from one model to the other very smooth and transparent.

The actors in SysML correspond to the main entities who interact with and influence the system through their actions [13] [14] [15]. This is very similar to the role of the participants in IBM Blockchain Platform, who are the business members that own assets in the system and can submit transactions that alter the system [31].

The core blocks in SysML are the main structural elements of the system, and where the process is built around of. These undergo several states during its lifecycle (detailed in the stm's) which change depending on the triggered event that happens during a process [13] [14] [15]. The assets in IBM Blockchain Platform are the goods or possessions being transacted from one participant to another within the system [31].

Events are the triggers for a block state change in a SysML model [13] [14] [15]. Transactions are the procedures, in IBM Blockchain Platform, that make an asset change ownership when certain criteria are met [31].

With its elements being very similar, the conversion of the SysML model to a Blockchain Model File was made, with the main actors becoming the participants, the core blocks becoming the assets and the events becoming the transactions. This conversion was made using the Hyperledger Composer Playground provided by the IBM Blockchain Platform [31]. Figure 22 illustrates this transition.

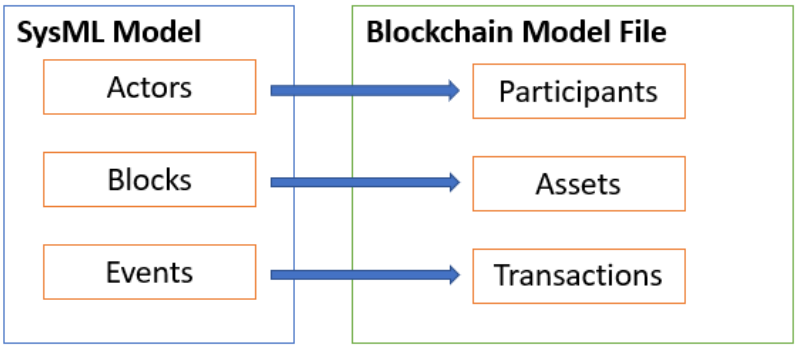


*Figure 22 - How the SysML and Blockchain model file elements were connected to each other*

# 4. Results & Discussion

This section of the project shows the resulting SysML and IBM Blockchain Platform models, built from the batch release process information and data provided by the Company. It also shows the testing for the IBM Blockchain Platform model using a sample population. Finally, it provides some commentary and what should be the future steps for this project.

## 4.1 SysML Model

### 4.1.1 Model Background

To find what makes a Batch Release and to help clarify what are the problem symptoms the Company is currently dealing with, there were eight meetings with the Company and one visit to the Pre-Wholesaler's installations (Annex B, page 62). These meetings resulted in several remarks about the Company's Batch Release Process:

- The Batch Release Process is a complex stage in the Company's supply chain, and there is no reference model which details the process steps and entities involved;
- There are two types of Batch Release: a central release which takes nine days and a local one which takes five days;
- After a shipment arrives, the Pre-Wholesaler has one day, established by contract, to register the batches in one of the Company's software;
- During the Batch Release there are two types of quality verification, one related to the quality of the product itself, and the other associated with the documentation about the product. Both the product and its documentation are always in constant update and need to comply with the Company's quality requirements and government regulations, which are different depending of the importing country;
- The value of the batches waiting to be released is approximately 5M €, meaning this is a big concern for the Company;
- The time logs were obtained from three of the several software platforms used by the Company in different steps of the Batch Release process.

A SysML model composed of nine diagrams was created taking these remarks in consideration.

### 4.1.2 Block Definition Diagram

The block definition diagram (Figure 114; Annex C, page 68) shows the structural elements of the Batch Release Process.

In Figure 23 the block Batch is defined. Associated with it is the Batch Reception block, which refers to the entry made about the batch in Software D. Each Batch can only have one Batch Reception and each entry can only have one batch associated. The Batch block is in one of 3 different conditions at the end of the release process: Released, Rejected or Discarded. These specialized blocks are represented in the bdd with an upwards arrow pointing to the generalized batch block. A Released Batch is a batch that has successfully passed quality control and is ready for the market. A Rejected

Batch is a batch that cannot be released due to having unredeemable faults detected by a Company Qualified Person (QP). A Discarded Batch is a batch that is corrupted, which means it cannot be released due to an obvious fault that was detected before the quality review.

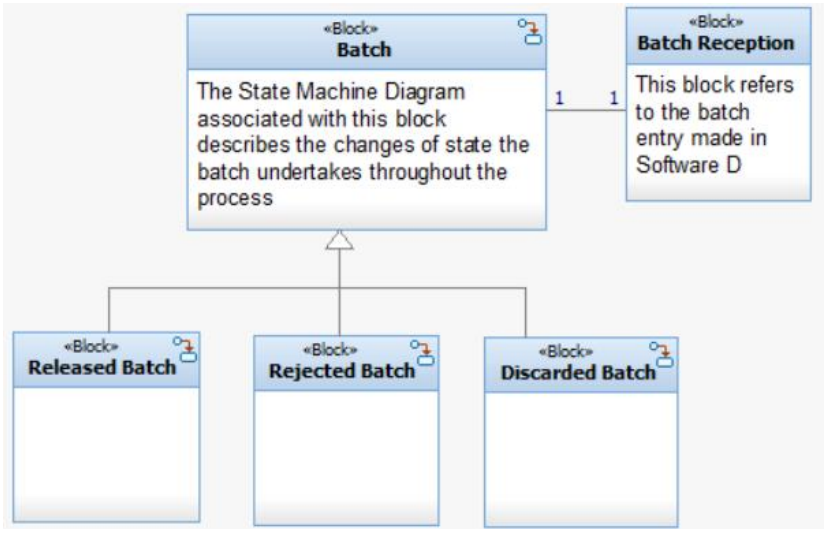


*Figure 23 - The Batch block*

Figure 24 depicts the Duty of Care (DoC) block with its associated properties and blocks. The DoC is the single document that details the condition and status of the Batch. Each DoC has associated with it five pieces of information: the temperature registry, the Certificate of Analysis (CoA), the European Union Batch Certificate (EUBC), the shipping advice and the product composition data. During the Central Release sub-process, the DoC will be a Pending DoC while it awaits approval from the European Union Qualified Person (QPEU).

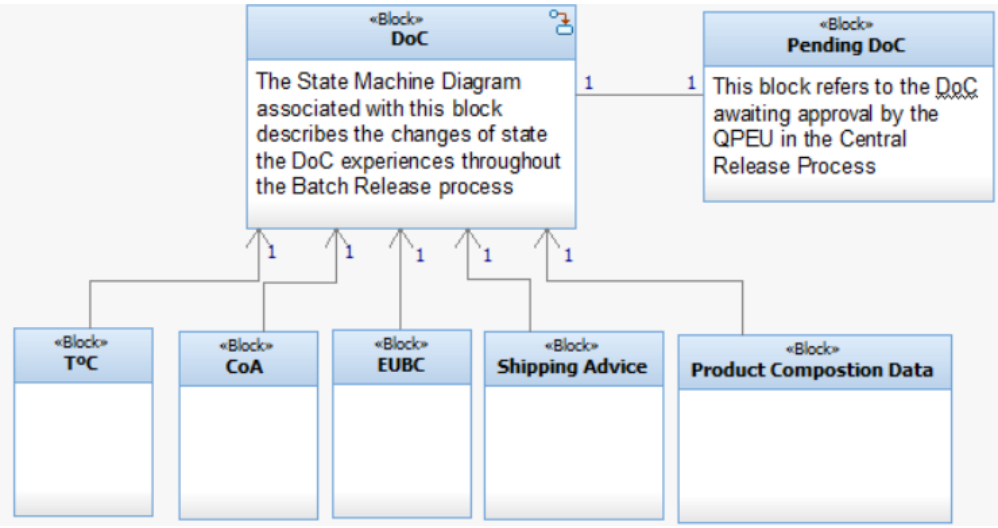


*Figure 24 - The DoC block*

Figure 25 identifies the two types of release sub-processes. The Local Batch Release is for batches that can be released by a Portuguese Qualified Person (QPPT) in Portugal, while the Central Batch Release is for batches that require the approval of a QPEU in the European Union.

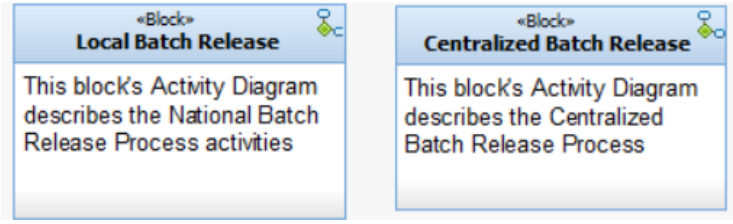


*Figure 25 - The two different process blocks*

Figure 26 defines the European Union Batch Release Certificate (EUBRC) and its local counterpart the Batch Release Certificate (BRC). The BRC is a required certificate that states the batch is eligible for release.

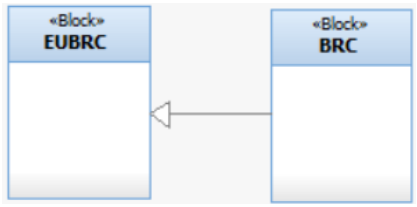


*Figure 26 - The BRC blocks*

#### *4.1.2.1 Block Definition Diagram Remarks*

- The Block Definition Diagram defines the core assets of the batch release process;
- The defined blocks represent the Batch, the DoC, the two types of release process and the BRC, along with the relevant structural elements which help define these blocks.

### 4.1.3 Use Case Diagram

The use case diagram (Figure 115; Annex C, page 68) depicts the relevant actors that interact with the process through its use cases.

Figure 27 shows the two actors outside of the company that interact with the process. The Incoming Truck Driver that delivers the batch shipments to the warehouse. And the Pre-Wholesaler's Warehouse, that performs a verification to the batch shipments that have arrived.

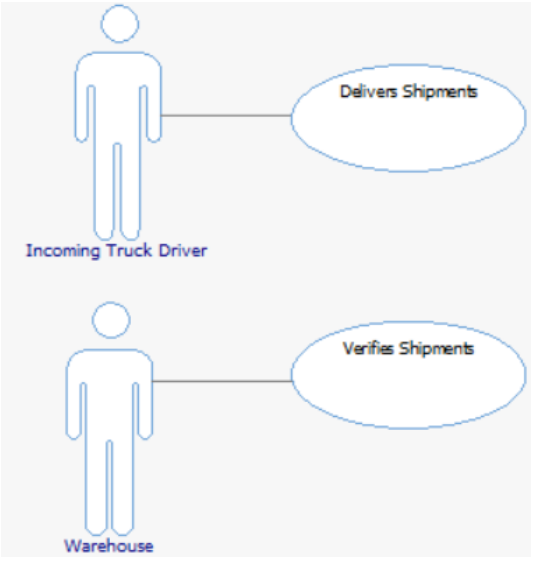


*Figure 27 - The actors external to the Company*

Figure 28 depicts the three actors inside the Company that interact with the process. The Quality Department which assesses the quality of the batch, the Logistics Department that places the buy orders of necessary batches to be delivered at the Pre-Wholesaler's warehouse, and the Software that supply and keep track of the batch records or other relevant information.

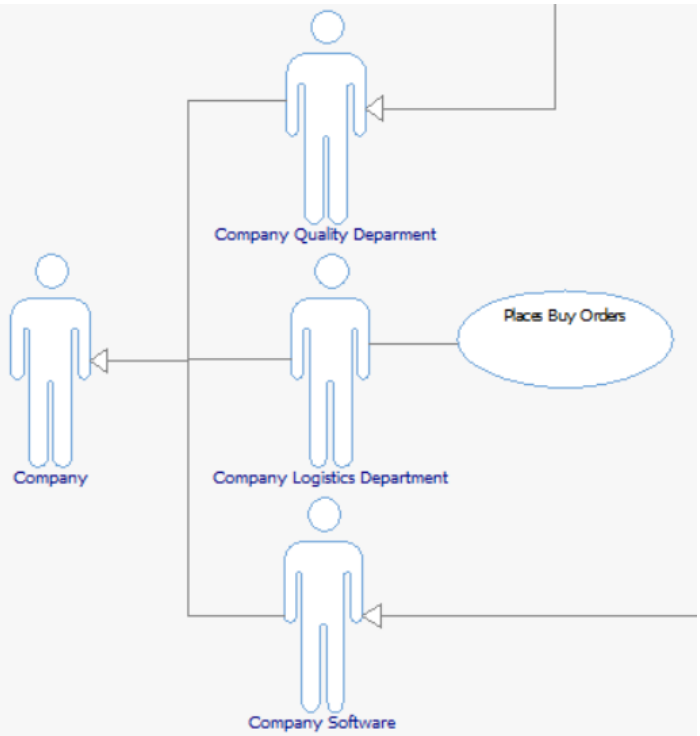


*Figure 28 - The actors inside the Company*

In Figure 29 the Company's Quality Department is illustrated. The Quality Assurance (QA) reviews the quality of the batches and assesses any faults detected. A specialized QA, the Qualified Person (QP), creates the BRC stating the batch eligibility for release. It worth noting that the main difference between both Company QP actors is that the QPPT is responsible for creating the BRC for the Local Release Process, while the QPEU creates the BRC for the Central Release Process after rechecking the batch's DoC.

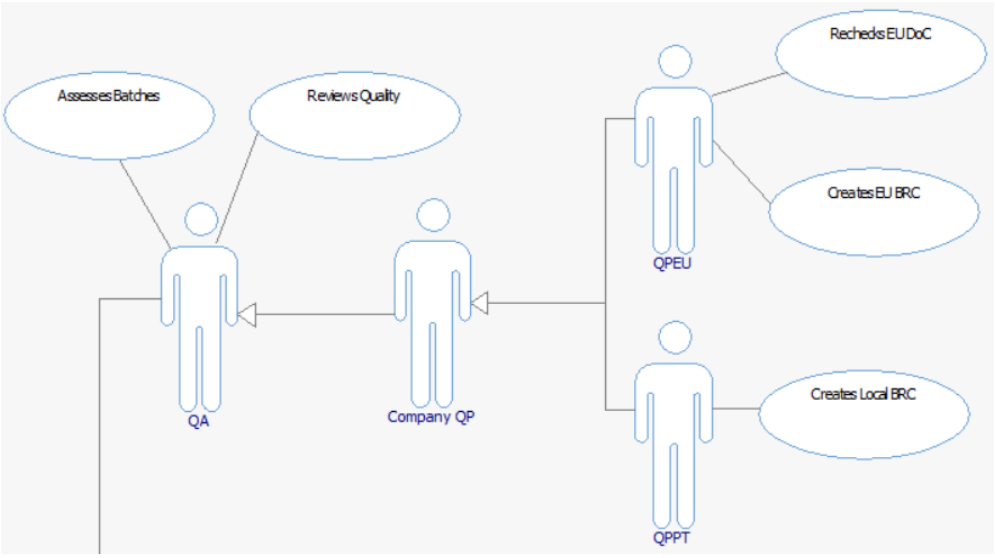


*Figure 29 - The Company's quality department*

Figure 30 shows the four different software products used by the Company during the Batch Release process. Software A keeps track of the temperature records during the transportation of the batch shipment to the Pre-Wholesaler's warehouse, while also supplying this information afterwards. Software B is a data base that supplies the product composition data for the quality assessment of the batch. Software C is a platform where the local quality department uploads batch verification records to be reviewed by the QPEU and then re-uploaded to be released locally. Software D is where the batch is registered with its details and current status.

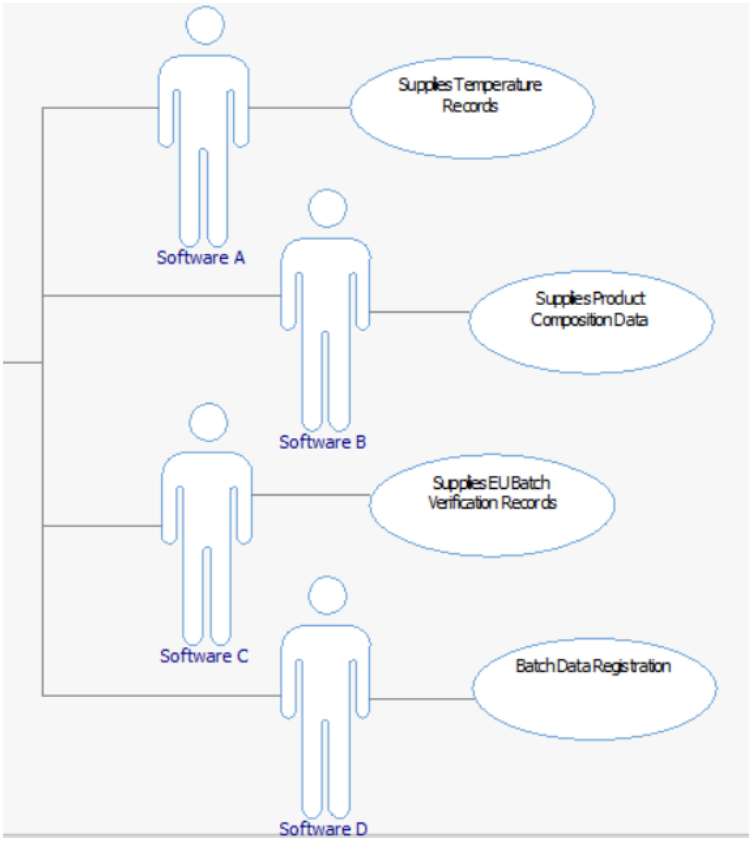


*Figure 30 - The Company's software actors*

#### *4.1.3.1 Use Case Diagram Remarks*

- The Use Case Diagram defines the main actors within the Company which interact with the batch release process;
- It also shows how the actors interact with the process through their actions;
- Some actors have very similar interactions with the batch release process. For example, Software A, B and C all store and supply information, although about different process subjects.

### 4.1.4 Activity Diagrams

#### *4.1.4.1 Local Release ACT*

The local release activity diagram (Figure 116; Annex C, page 69), shows the order of the actions of the local process and the related inputs and outputs.

As shown in Figure 31, the process begins with the arrival of the shipment and its data at the Pre-Wholesaler's warehouse. A shipment verification is performed resulting in a registry that creates both the batch Duty of Care (DoC) and the batch reception entry in Software D, evidenced in Figure 32. The warehouse fills its section in the DoC, which now contains warehouse data.

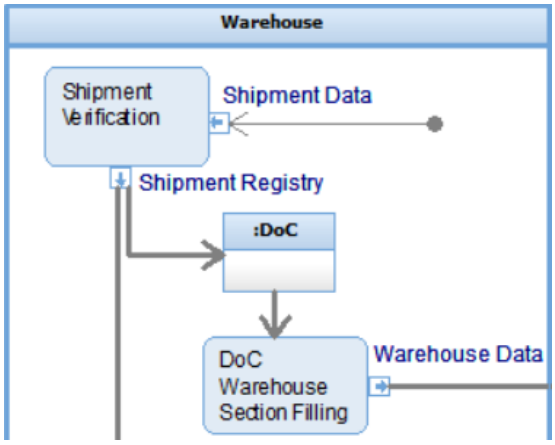


*Figure 31 - Warehouse flow of actions*

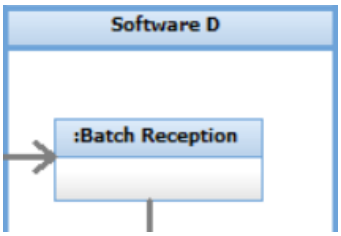


*Figure 32 - Batch Reception entry in Software D*

Figure 33 shows the DoC quality section filling. The QPPT uses the warehouse section data, along with the CoA, EUBC, Shipping Advice, Temperature Records (supplied by Software A) and the results of the artworks and Software B aided review to fill the quality section of the DoC.

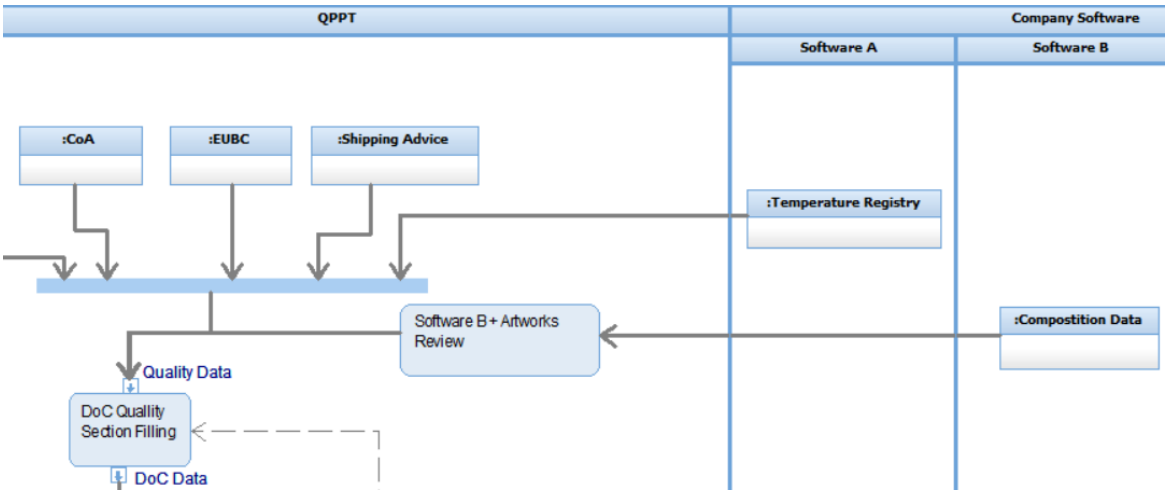


*Figure 33 - DoC quality section filling and its inputs*

As evidenced in Figure 34, the QPPT performs several quality checks using information in the DoC. In the first check, if any crucial piece of information associated with the batch is missing (e.g. if there is not a CoA, the batch cannot be released), the batch is deemed corrupted and is disposed of, resulting in a Discarded Batch. This is shown in Figure 35.

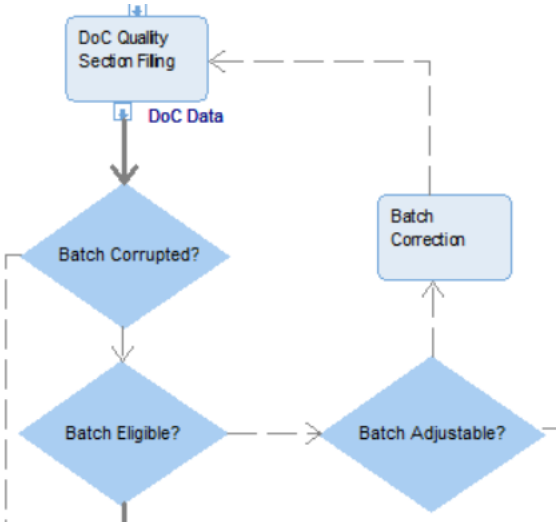


*Figure 34 - The quality checks performed by the QPPT*

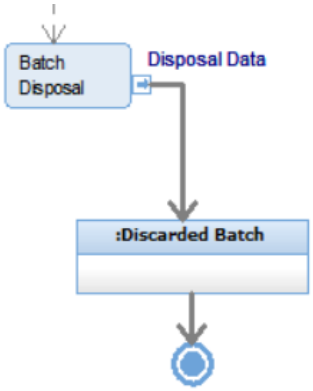


*Figure 35 - Disposal of a corrupted Batch*

If the batch is not corrupted, the batch eligibility is assessed. This means the appraising the faults detected in the batch. If it is not eligible for release, another check is made to see if the batch can be adjusted to correct these faults. If the batch can be altered, the DoC is changed to reflect these modifications. Otherwise, it's classified as unredeemable and rejected, resulting in a Rejected Batch, as illustrated in Figure 36.

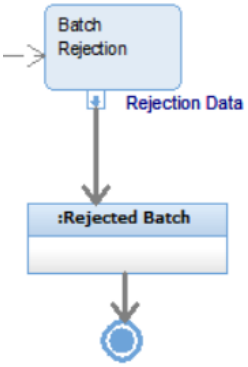


*Figure 36 - Rejection of an unredeemable Batch*

If the batch is eligible for release, the QPPT creates a BRC and updates the batch status in Software D, resulting in a Released Batch. As shown in Figure 37, the BRC is used as an input to update the batch status in Software D, which, illustrated in Figure 38, releases the batch.

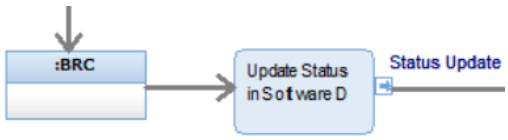


*Figure 37 - Updating the Batch status in Software D*

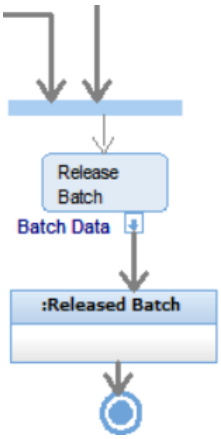


*Figure 38 - Releasing the Batch*

### 4.1.4.2 Central Release ACT

The central release activity diagram (Figure 117; Annex C, page 70) has a few differences from its local counterpart.

As evidenced in Figure 39, the quality review and the first quality check are performed by a QA instead of a specialized QPPT, due to this actor being in very few numbers. When the batch check is completed, and its results are acceptable, the DoC data is uploaded to Software C, where it waits the review from the QPEU, as seen in Figure 40.

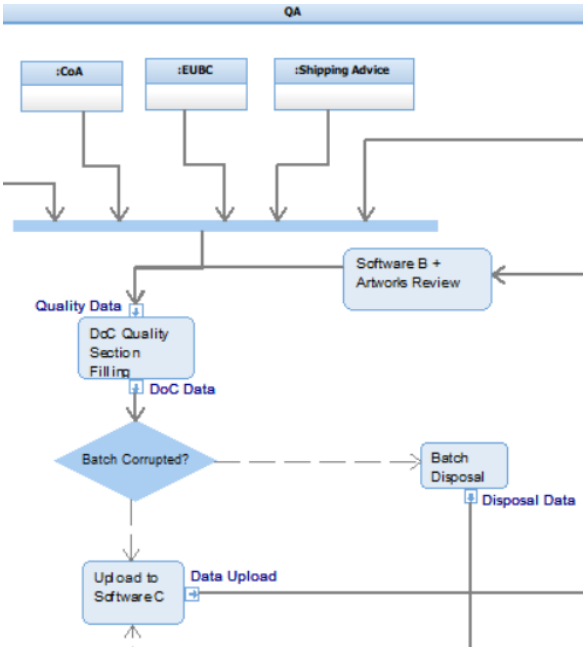


*Figure 39 - The general QA performing the quality review*

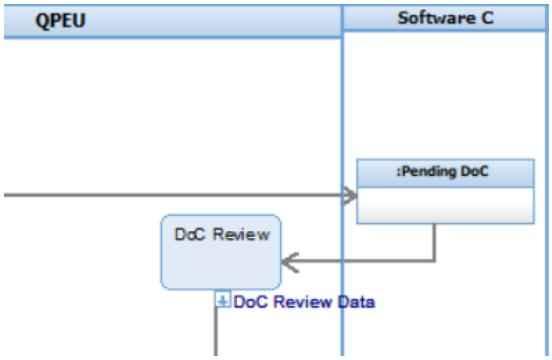


*Figure 40 - The Pending DoC*

As shown in Figure 41 When this review is completed, the QPEU performs the other two checks, verifying the batch's eligibility and adjustability in case faults were detected. If eligible, the QPEU uploads the EUBRC to Software C.

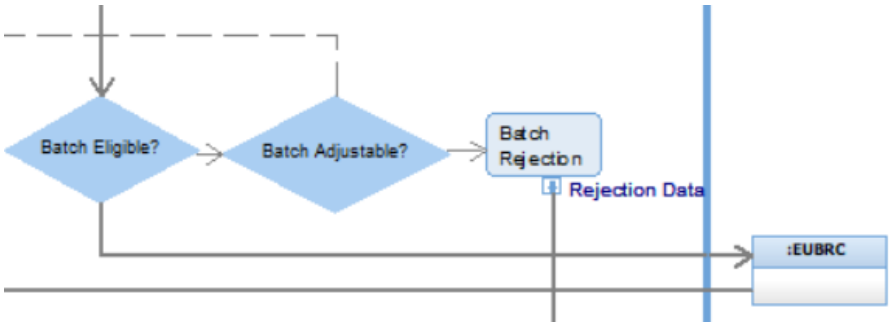


*Figure 41 - The QPEU creating the EUBRC*

When the local QA receives this data, the batch status is updated in Software D, seen in Figure 42, and the batch is released.

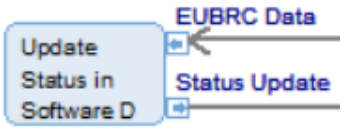


*Figure 42 - The update batch status action performed by the QA*

#### 4.1.4.3 Activity Diagram Remarks

- The Activity Diagrams show the order of the actions of both local and central batch release process and which actor executes them;
- The central release process includes most of the activities of the local release process, although performed by a different actor, a QA instead of a QPPT;
- Software C and the QPEU are only present in the central release process activities.

### 4.1.5 State Machine Diagrams

#### *4.1.5.1 Batch STM*

The batch state machine diagram (Figure 118; Annex C, page 71) describes the triggers and changes of state the batch goes through until the end of the release process.

As depicted in Figure 43 the batch life cycle starts at its creation, represented by a grey dot and an arrow pointing to the initial state, named Batch Created. When it arrives at the Pre-Wholesaler's warehouse its classified as Batch Arrived. In this state the batch is being verified by the warehouse until being registered in Software D, where it now is a Verified Batch. From this state there are three possible final states, symbolized by an outwards arrow from the end state pointing to a blue double circle, as evidenced in Figure 44.

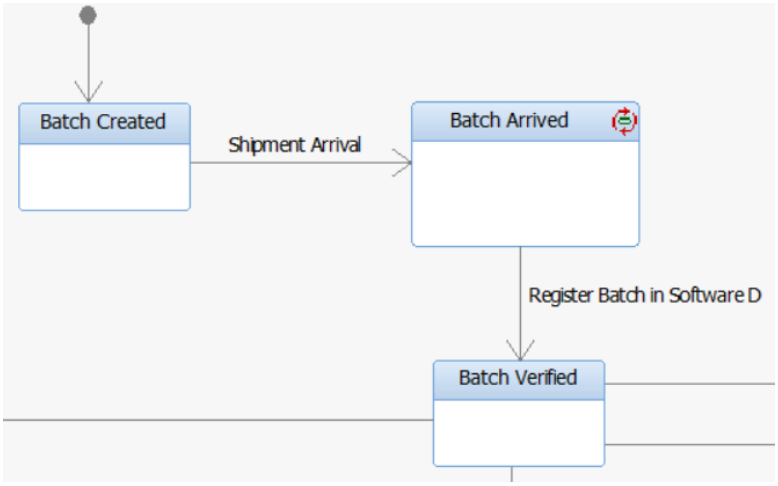


*Figure 43 - The first three states of the Batch*

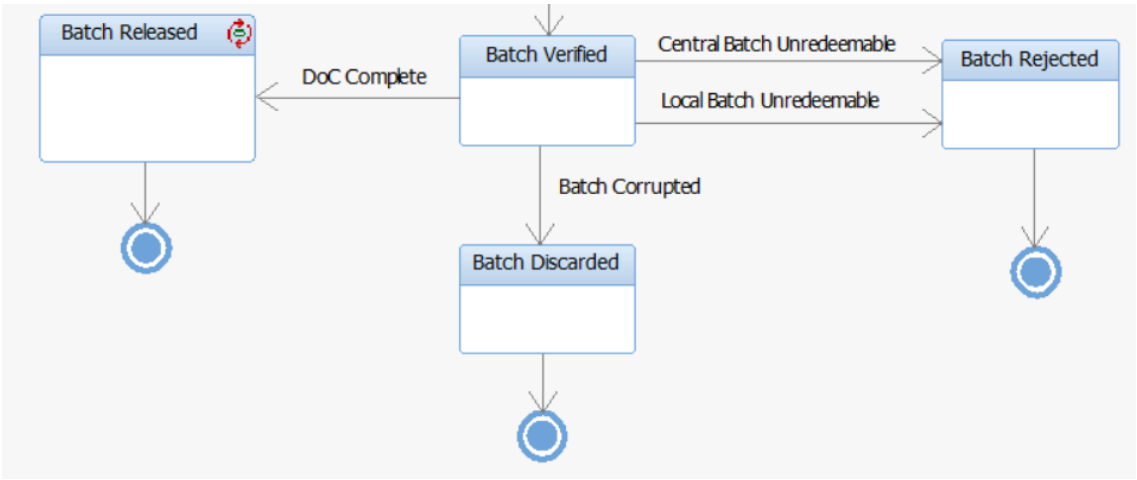


*Figure 44 - The three final states of the Batch*

If the associated batch DoC is successfully completed, the batch is deemed released. If the batch is classified as unredeemable due to its faults (this can occur in either release process), the batch is deemed rejected. If the batch is classified as corrupted due to missing a piece of vital information required for the quality review, the batch is deemed discarded.

#### 4.1.5.2 DoC STM

The DoC state machine diagram (Figure 119; Annex C, page 71) shows the triggers and different states the DoC has until the end of the release process. Recalling, each batch has an associated DoC.

Figure 45 illustrates the first states of the DoC. It starts blank until the batch is registered in Software D, where its header is automatically filled. When the Pre-Wholesaler's completes its section, it awaits a quality review by the QA or QPPT, depending on the release process associated with the batch.

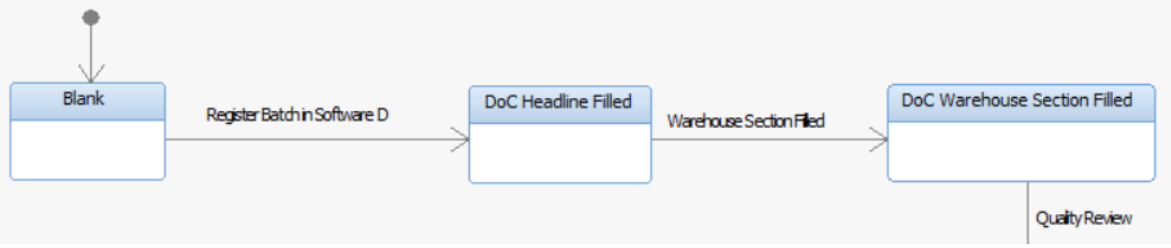


*Figure 45 - The first states of the DoC*

As seen in Figure 46 when the DoC quality section is filled, the next state depends on the type of release process, unless the batch is deemed corrupted by the QA, resulting therefore in a Discarded state. If the batch is released locally, the DoC is classified as nearly complete by the QPPT. If released centrally, illustrated in Figure 47, the batch is classified as a central release DoC and its uploaded to Software C to be reviewed by the QPEU.

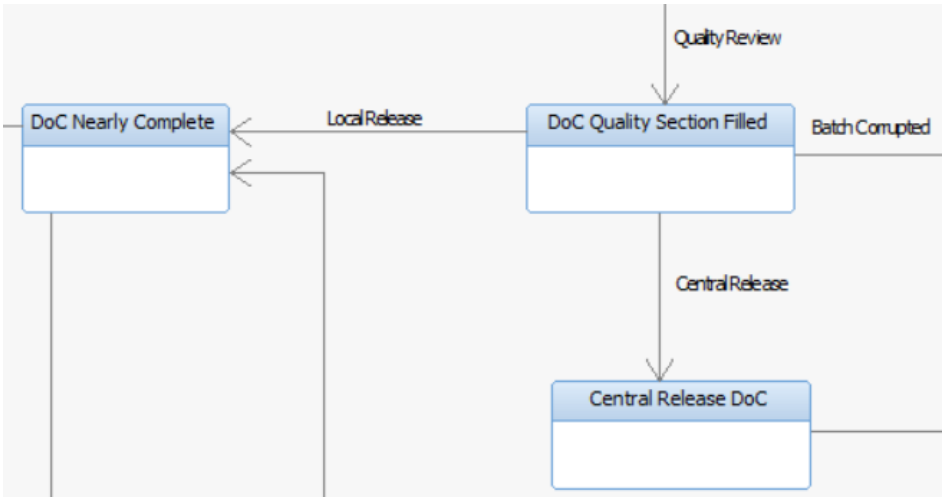


*Figure 46 - The Quality Section Filling output states*

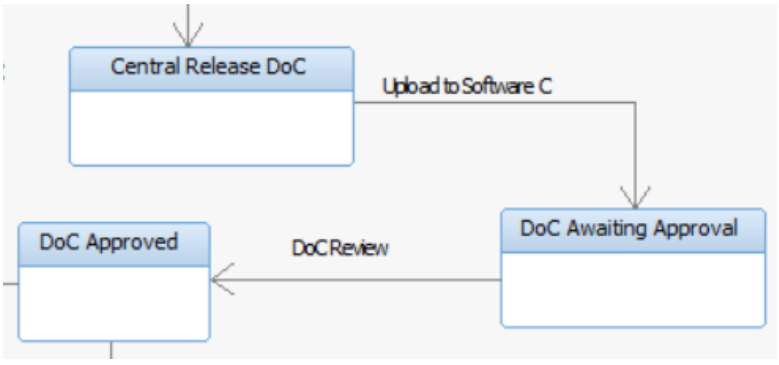


*Figure 47 - A centrally released DoC*

Once the DoC review is complete the DoC is considered approved. When in this state, if the respective batch contains unredeemable faults the DoC is rejected. Otherwise the DoC is updated in Software C and is classified as nearly complete, as evidenced in Figure 48.

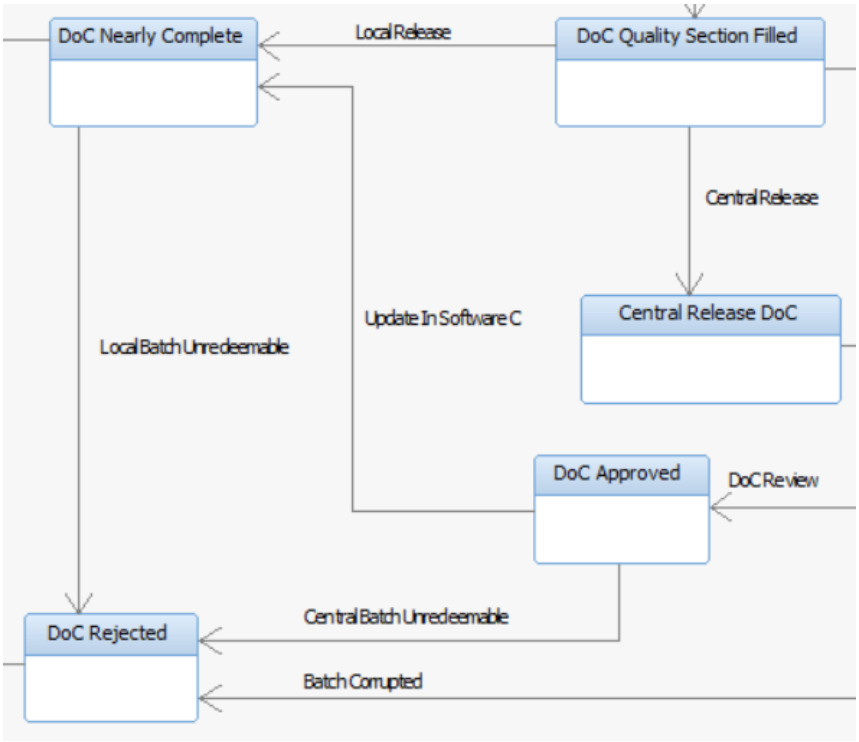


*Figure 48 - The input and output states of Doc Nearly Complete*

For locally released batches, if its DoC is nearly complete, but the batch contains unredeemable faults, the DoC is rejected. After updating the batch status in Software D, as shown in Figure 49, the DoC is considered complete ending its lifecycle.

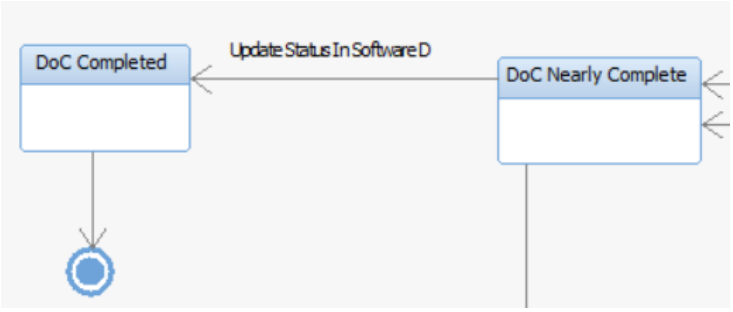


*Figure 49 - The DoC Completed State*

#### *4.1.5.3 State Machine Diagram Remarks*

- The State Machine Diagrams show the state changes of the two core elements of the batch release process, the Batch and the DoC, during their lifecycle;
- Both stm's have some identical triggers, meaning the state change is achieved at the same moment for both the Batch and DoC. For example, when Batch Corrupted trigger happens, the Batch changes to Discarded Batch and the DoC to DoC Rejected, simultaneously;

- The terminal states for both the central and local release batch DoC are the same, although the central release requires additional stage changes up to the final states.

### 4.1.6 Variables

To evaluate the current performance of the batch release process, a block definition diagram containing variables was created (Figure 121; Annex C, page 73).

Seen in Figure 50 is the deviations found variable. It was created to keep track of how many faulty batches were detected during the batch release process for a certain period (e.g. faults per year).

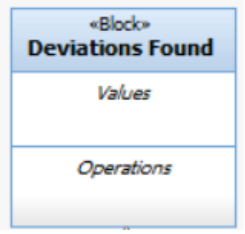


*Figure 50 - The deviations found variable*

The deviations found by the QPEU, QA and Warehouse, shown in Figure 51 are the faulty batches detected by each of these entities during the release process. As there are no other detection points in the system, the sum of the value of these variables must add up to the number of deviations found.

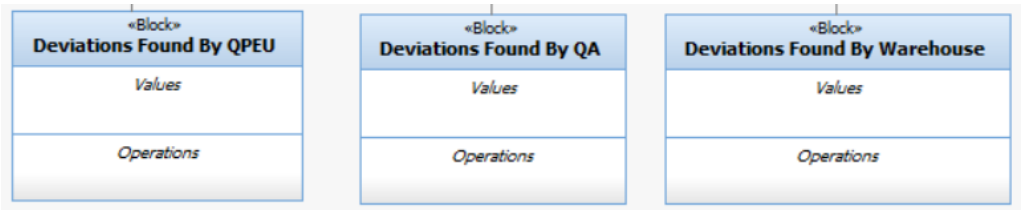


*Figure 51 - The deviations found by each of the entities of the process*

The number of rejected batches corresponds to the portion of deviated batches found that cannot be redeemed and were excluded. The corrected batches are the faulty batches that were adjusted to become acceptable. The sum of these two variables must also add up to the value of the deviations found variable. These variables are represented in Figure 52.


*Figure 52 - The resultant statuses of the faulty batches*

Illustrated in Figure 53, the variable QPEU workload states the amount of batch DoC's the QPEU verifies during a certain period. This number corresponds to the centrally released batches as the local release process does not require a QPEU verification. This variable is useful for assessing the need of the QPEU in the central release process, as this entity is a valuable and scarce resource and might be needed in another process.

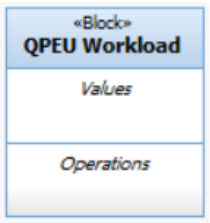


*Figure 53 - The QPEU workload variable*

#### *4.1.6.1 Variables Remarks*

- Variables support the appraisal of the current performance of the Company's batch release process;
- They measure the outputs of certain actions performed during the batch release process;
- These can be later used to create Key Performance Indicators (KPI's) allowing the Company to act according the behavior observed.

### 4.1.7 Sequence Diagrams

#### *4.1.7.1 Batch SD*

The batch sequence diagram (Figure 120; Annex C, page 72) describes the sequence of events and actions that are directly related with the batch.

The truck driver delivers the batch shipment at the Pre-Wholesaler's warehouse, as shown in Figure 54. This shipment also contains the associated batch data.

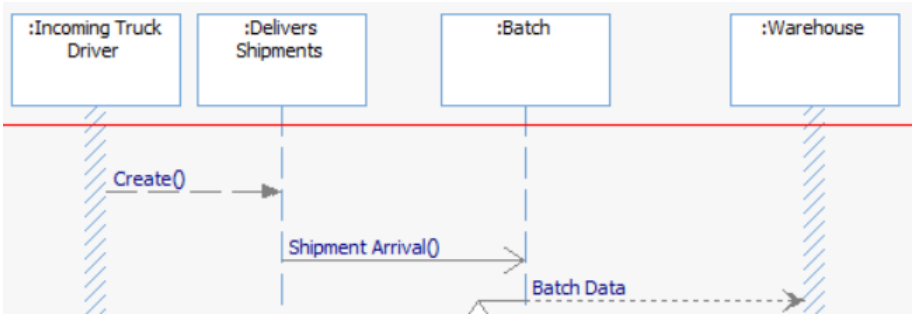


*Figure 54 - The arrival of the batch at the Pre-Wholesaler's warehouse*

The warehouse performs a batch verification after its arrival, as seen in Figure 55. The agreement between the Company and the Pre-Wholesaler dictates this task must be performed within one day at maximum after the shipment arrival.

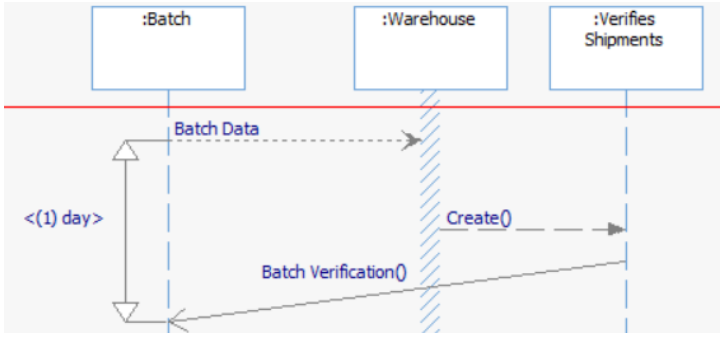


*Figure 55 - The warehouse verifies the batch*

After the warehouse verification the batch is registered in Software D, as represented in Figure 56. Afterwards the Company will proceed to complete the batch DoC to release the batch. This stage takes seven days for the central release process and four for the local release.

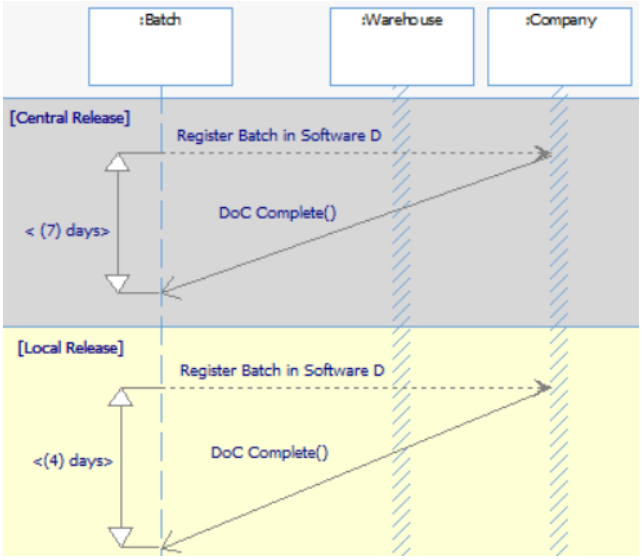


*Figure 56 - Comparison between the Central and Local Release Process duration*

### *4.1.7.2 DoC SD*

The DoC sequence diagram (Figure 122; Annex C, page 74) shows the sequences of events and actions related to the batch DoC.

As seen in Figure 57, when the batch shipment arrives at the Pre-Wholesaler's warehouse Software A creates an entry, with a time stamp, showing the last temperature recorded during the transportation.

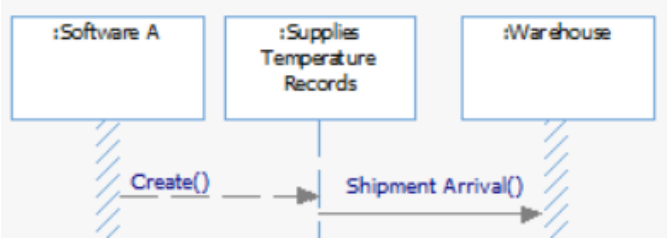


*Figure 57 - Software A creating a temperature record*

After the shipment arrival, the warehouse proceeds with a shipment verification, which may increase the value of deviations found by the warehouse if any faults were detected, as evidenced in Figure 58.

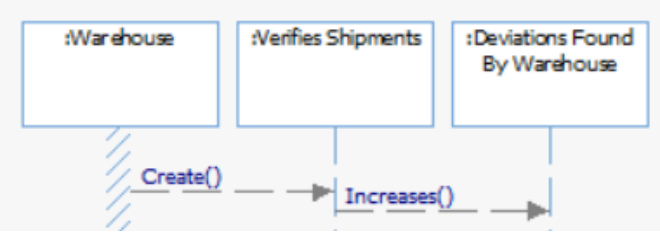


*Figure 58 - Warehouse verifying the shipment*

When the batch verification is completed, as represented in Figure 59 the warehouse fills its corresponding section of the DoC with the relevant information, including the deviations found, and registers the batch in Software D.

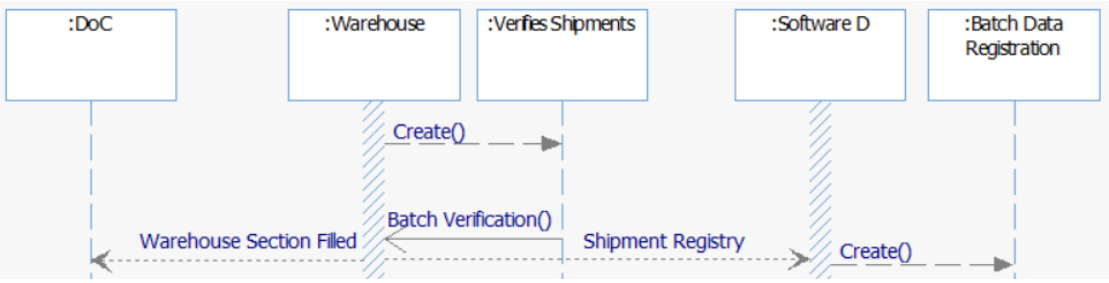


*Figure 59 - Warehouse registering the shipment in Software D*

The Company's temperature records from Software A and batch entries from Software B, were used to calculate how long this process stage takes to be completed, resulting in three days, as seen in Figure 60. This is a larger period than the one-day time limit that was agreed with the Pre-Wholesaler beforehand.

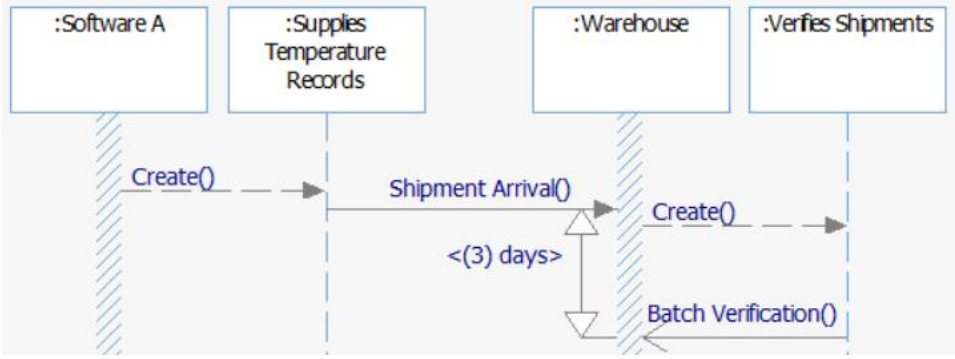


*Figure 60 - The period between the shipment arrival and when the verification is complete*

Depending on the process, a different entity will be responsible for executing the proceeding action. In the central release process, represented in Figure 61, after the batch is verified, the QA will perform a quality review, which may increase the value of the variable deviations found by QA. When the quality review is complete the QA fills the quality section of the DoC.

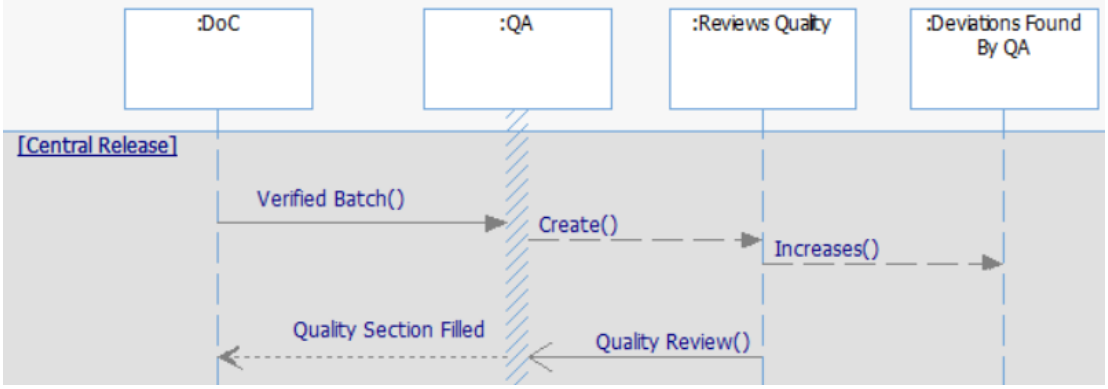


*Figure 61 - The quality review performed by the QA*

As evidenced in Figure 62, when the quality section is filled, the QA uploads the DoC data to Software C. This platform records this information to be later accessed by the QPEU.

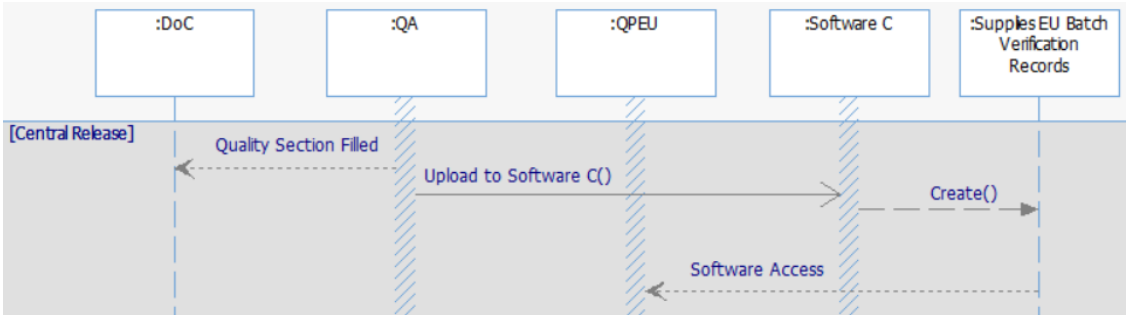


*Figure 62 - Software C keeping the DoC records*

After accessing the data in Software C, as shown in Figure 63, the QPEU proceeds with the review of the DoC, which may increase the number of deviations found by this entity. When the DoC review is complete, the QPEU updates the respective data in Software C so that it can be later accessed by the QA.

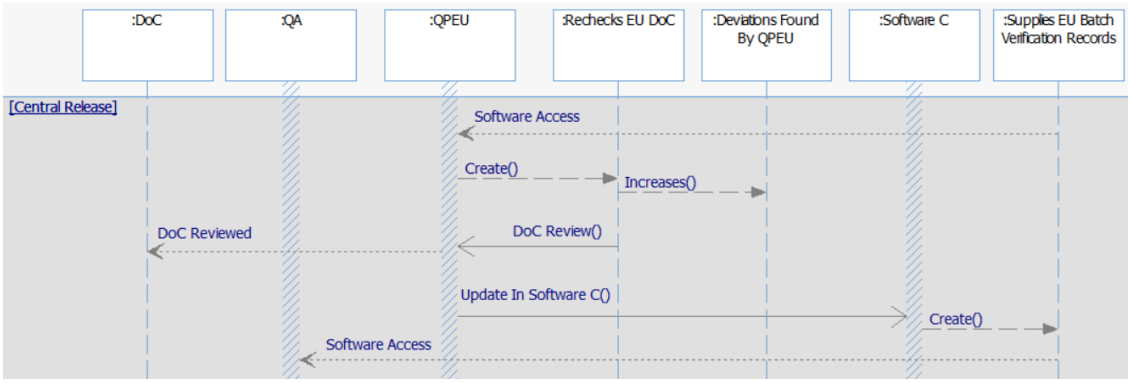


*Figure 63 - The QPEU reviewing the DoC*

As represented in Figure 64, using the newly uploaded data from Software C, the QA updates the batch status in Software D, completing the DoC and releasing the corresponding batch.

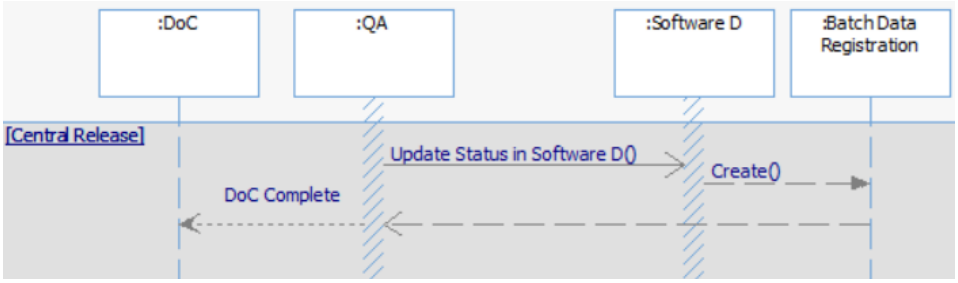


*Figure 64 - The QA updating the batch status in Software D*

For the local release process, as illustrated in Figure 65, after the batch is verified, the QPPT performs the quality review of the batch. When this task is complete the QPPT fills the quality section of the DoC.

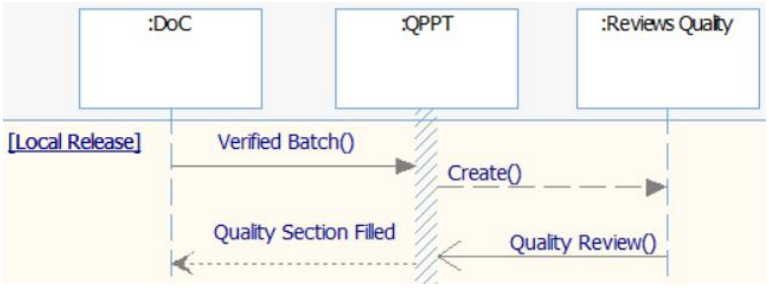


*Figure 65 - The quality review performed by the QPPT*

After the quality section is completed, as shown in Figure 66 ,the QPPT updates the batch status in Software D, releasing the batch and completing the DoC.

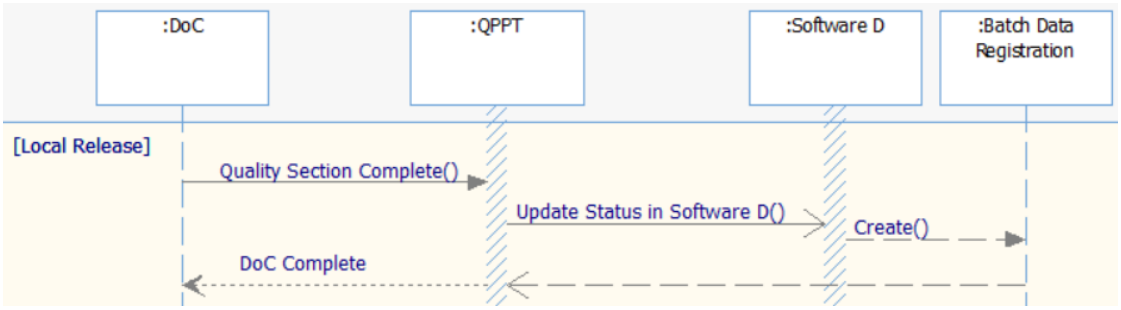


*Figure 66 - The DoC is complete after the batch status is updated in Software D*

Using the Software entries provided by the Company, the time spent, in the central release process, between the warehouse batch registry in Software D and the QA uploading data in Software C was calculated, resulting in four days, represented in Figure 67. This matches the period, in the local release process, between the warehouse batch registry and the QPPT updating the batch status in Software D, shown in Figure 68.

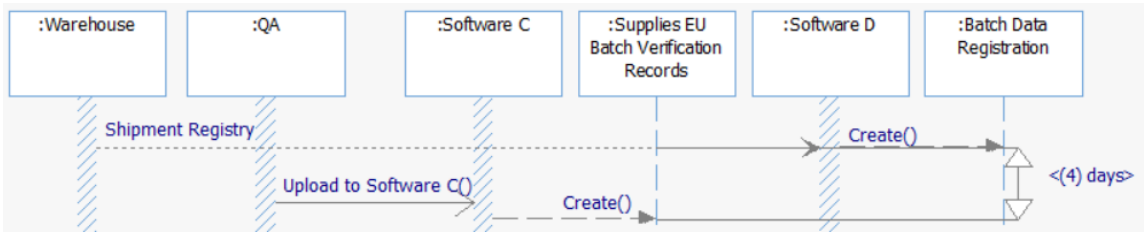


*Figure 67 - The period between the warehouse batch registry and the QA uploading the DoC to Software C*

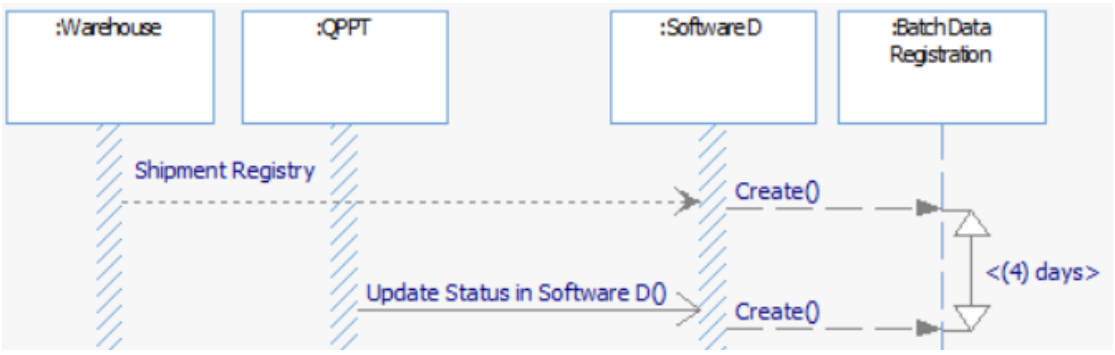


*Figure 68 - The period between the warehouse batch registry and the QPPT updating the status in Software D*

The data entries the Company provided were also used to calculate the period between the QA uploading the DoC to Software C and the QPEU updating this information, which resulted in three days, as seen in Figure 69. The time between this update and the QA updating the batch status in Software D is negligible (about 8 minutes).

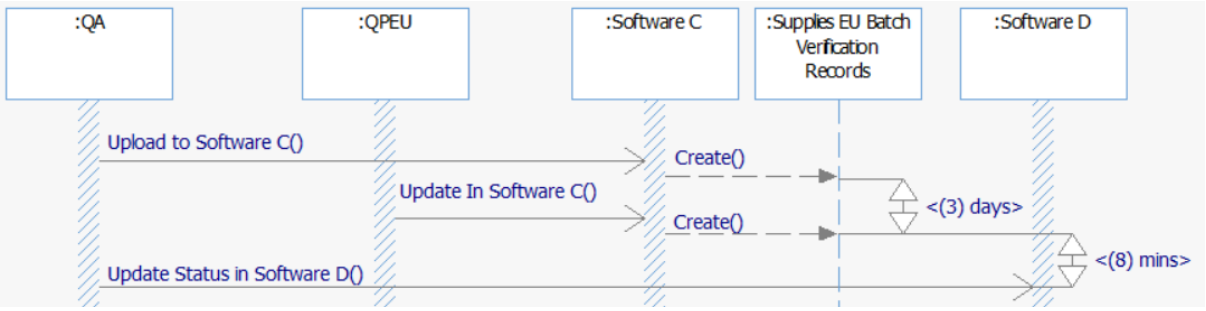


*Figure 69 - The period the QPEU requires to perform its allocated tasks*

As illustrated in Figure 70, joining the periods of the two process stages inside the Company means the central batch release process takes seven days within the Company itself. Adding the three days from the Pre-Wholesaler's warehouse batch verification, results in the central release process having ten days total.

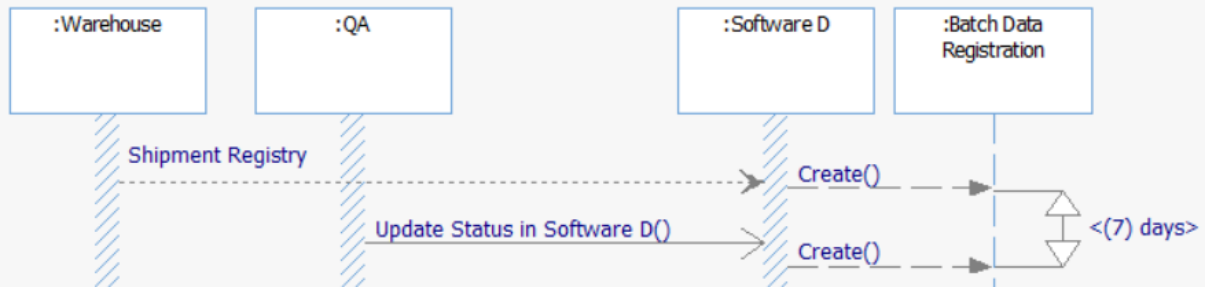


*Figure 70 - The period the Company actors spend to centrally release a batch*

#### *4.1.7.3 Sequence Diagram Remarks*

- The Sequence Diagrams show the order of the actions performed by the actors during the batch release process and the effect these have on the system;
- Using the timestamps obtained from software data logs, the diagrams also show the duration of some stages of the process;
- How the variables change value is also illustrated with these diagrams.

### 4.1.8 Remarks about the SysML model

After building the SysML model, several observations about the Company batch release process were made:

- The Pre-Wholesaler's warehouse is taking three days to perform the batch verification, not meeting the agreement to complete this task in one day;
- The central batch release process takes three days more than the local one, due to the need of a DoC review performed by the QPEU;
- There is only information about the number of rejected batches and not how many faults were found and by whom, meaning there's no way to check the actual need for each step, because each actor's actual effectiveness is unknown.

## 4.2 Blockchain Model

Using the SysML model as the base architecture, a Blockchain business model file (.cto) was built in Hyperledger Composer Playground.

### 4.2.1 Participants

One abstract participant, shown in Figure 71, was modeled to serve as the base for the other three participants, since all of them have these common attributes: e-mail, username and last name. The username is the unique identifier that sets apart each participant.

```
/**
 * An Actor participant
 */
abstract participant Actor identified by username {
  o String email
  o String username
  o String lastName
}
```

*Figure 71 - The model file code for the abstract Actor participant*

It's worth noting that, if additional modifications for the attributes of any entity are needed, these can be easily added to the model file.

The three participants were modeled after the main actors involved with the batch release process: Warehouse, QA and QPEU.

The warehouse participant, seen in Figure 72, besides the attributes inherited from the abstract actor participant, has an address, county and postcode attributes. Figure 73 shows how a participant is created in the software by filling the required attributes with the relevant information.

```
/**
 * A Warehouse participant.
 */
participant Warehouse extends Actor {
    o String address
    o String county
    o String postcode
}
```

*Figure 72 - The model file code for the warehouse participant*

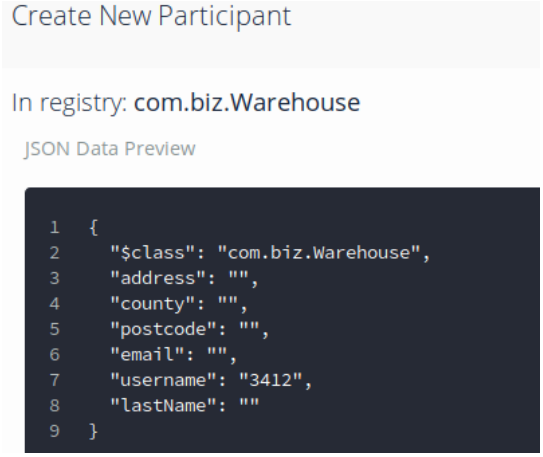


*Figure 73 - How a warehouse participant is created*

The QA participant, represented in Figure 74, extends the attributes of the abstract actor participant adding the address, county and postcode. A fourth attribute defines the position the QA has.

```
/**
 * A QA participant.
 */
participant QA extends Actor {
    o String address
    o String county
    o String postcode
    o Qposition position
}
```

*Figure 74 - The model file code for the QA participant*

As seen in Figure 75, the QA can either be a general QA or a more specific QPPT. This is important to define since some actions require a QPPT instead of a general QA, preventing unauthorized activities.

```
/**
 * The two positions within the Quality Department of the Company
 */
enum Qposition {
  o QA
  o QPPT
}
```

*Figure 75 - The model file code for the QA positions*

The QPEU participant, demonstrated in Figure 76, besides the attributes inherited from the abstract actor participant, has the attribute country. This is because the QPEU might be from another country than the QA involved with the batch release process.

```
/**
* A QPEU participant.
*/
participant QPEU extends Actor {
  o String country
  o String address
  o String county
  o String postcode
}
```

*Figure 76 - The model file code for the QPEU participant*

#### *4.2.1.1 Participant Remarks*

- The participants are the entities within the Company's actors which submit transactions in the batch release process Blockchain business network;
- These were modeled after the main actors of the batch release process SysML model;
- These are the only entities with access to the Blockchain business network records and can submit transactions, although, if required, this can be altered.

### 4.2.2 Assets

The two assets were modeled after the most relevant blocks in the process: the batch and the DoC, both defined in the process block definition diagram in SysML.

The batch asset, represented in Figure 77, has the attributes BatchNum, BatchState, BatchType and origin, and its identifier is the BatchNum which corresponds to the batch number.

```
/**
 * A Batch asset.
 */
asset Batch identified by batchNum {
    o String batchNum
    o BatchState batchState
    o BatchType batchType
    o String origin
}
```

*Figure 77 - The model file code for the batch asset*

The BatchState attribute, seen in Figure 78, shows the current state of the batch in its lifecycle while the BatchType, presented in Figure 79, shows the type of batch. The batch states were obtained from the batch state machine diagram in the SysML model. If additional types are created these can be easily implemented in the model file by adding the respective batch types in the code.

```
/**
 * The different States a Batch goes through during its lifecycle.
 */
enum BatchState {
  o Batch_Created
  o Batch_Arrived
  o Batch_Verified
  o Batch_Released
  o Batch_Discarded
  o Batch_Rejected
}
```

*Figure 78 - The model file code for BatchState attribute*

```
/**
 * The several types of Batches
 */
enum BatchType {
  o Pharmaceutical
  o Medical
  o Psychotropics
  o Other
}
```

*Figure 79 - The model file for BatchType attribute*

The DoC asset, shown in Figure 80, has the attributes BatchNum, DoCState and Batch, with its identifier being the BatchNum. Because the DoC is associated with a batch, its BatchNum identifier needs to match its respective Batch identifier, an attribute with the same name.

```
/**
 * A DoC asset. A DoC is the information document related to a Batch
 */
asset DoC identified by batchNum {
  o String batchNum
  o DoCState docState
  --> Batch Batch
}
```

*Figure 80 - The model file for the DoC asset*

The DoCState attribute, represented in Figure 81, shows the current state of the DoC in its lifecycle. These states were obtained from the DoC state machine diagram.

```
/**
 * The different States a DoC goes through during its lifecycle.
 */
enum DoCState {
  o Blank
  o DoC_Headline_Filled
  o DoC_Warehouse_Section_Filled
  o DoC_Quality_Section_Filled
  o Central_Release_DoC
  o DoC_Awaiting_Approval
  o DoC_Approved
  o DoC_Rejected
  o DoC_Nearly_Complete
  o DoC_Completed
}
```

*Figure 81 - The model file for the DoCState attribute*

#### *4.2.2.1 Asset Remarks*

- The assets were modeled after the Batch and DoC, the core blocks defined in the SysML model and the elements on which the batch release process is centered;
- The attributes BatchState and DoCState were modeled after the states represented the respective state machine diagrams;
- If required by the Company, more attributes can be added to the assets.

### 4.2.3 Transactions

Most of the transactions in the business network were modeled after the events present in the state machine diagrams of the batch release process SysML model and were organized in two groups: related to the batch asset and related the DoC asset.

It's important to note that, for transactions to occur, the user input in the attributes modeled in the transaction need to match the system requirements displayed in the script file (.js). Otherwise, an error message stating why the transaction cannot occur should be shown in response. Valid entities, Asset or Participant, are those defined by the user before submitting a transaction. Entities not defined beforehand are not considered valid.

Two abstract transactions, the BatchStateChange, shown in Figure 82, and DoCStateChange, seen in Figure 83, were created to identify the corresponding asset being transacted, the Batch and DoC asset respectively. All other transactions were created extending either of these two, depending on the associated asset.

```
/* An abstract transaction type for Batch State Change movements.
 */
abstract transaction BatchStateChange {
    --> Batch Batch
}
```

*Figure 82 - The model file for the abstract transaction BatchStateChange*

```
/* An abstract transaction type for DoC State Change movements.
 */
abstract transaction DoCStateChange {
    --> DoC DoC
}
```

*Figure 83 - The model file for the abstract transaction DoCStateChange*

The transaction ShipmentArrival, represented in Figure 84, occurs when the shipment arrives at the warehouse. It requires the user to input the Batch asset identifier, BatchState and the Warehouse participant identifier. For this transaction to occur the Batch asset should be valid, have the Batch_Created BatchState and a valid Warehouse participant. When it occurs, it results in the Batch asset changing its BatchState to Batch_Arrived.

```
transaction ShipmentArrival extends BatchStateChange {
  --> Batch BatchState
  --> Warehouse Warehouse
}
```

*Figure 84 - The model file for the transaction ShipmentArrival*

BatchVerified, demonstrated in Figure 85, happens when the batch is registered in Software D by the Warehouse. It requires a valid Batch asset with the BatchState Batch_Arrived and a valid Warehouse participant. When complete, this transaction results in the Batch asset having the BatchState Batch_Verified.

```
transaction BatchVerified extends BatchStateChange {
  --> Batch BatchState
  --> Warehouse Warehouse
}
```

*Figure 85 - The model file for the transaction BatchVerified*

DoCComplete, shown in Figure 86, occurs when the batch's DoC is classified as complete by either the QA or QPPT. It requires a valid Batch asset with the BatchState Batch_Verified and a valid QA participant. When complete, this transaction results in the Batch asset having the BatchState Batch_Released.

```
transaction DoCComplete extends BatchStateChange {
  --> Batch BatchState
  --> QA QA
}
```

*Figure 86 - The model file for the transaction DoCComplete*

BatchDiscarded, seen in Figure 87, happens when the batch is classified as corrupt by the QA or QPPT. It requires a valid Batch asset with the BatchState Batch_Verified and a valid QA participant. When complete, this transaction results in the Batch asset having the BatchState Batch_Discarded.

```
transaction BatchDiscarded extends BatchStateChange {
  --> Batch BatchState
  --> QA QA
}
```

*Figure 87 - The model file for the transaction BatchDiscarded*

CentralBatchRejected, represented in Figure 88 occurs when the batch, in a central release process, is classified as unredeemable by the QPEU. It requires a valid Batch asset with the BatchState Batch_Verified and a valid QPEU participant. When complete, this transaction results in the Batch asset having the BatchState Batch_Rejected.

```
165  transaction CentralBatchRejected extends BatchStateChange {
166    --> Batch BatchState
167    --> QPEU QPEU
168  }
```

*Figure 88 - The model file for the transaction CentralBatchRejected*

LocalBatchRejected, demonstrated in Figure 89, happens when the batch, in a local release process, is classified as unredeemable by the QA or QPPT. It requires a valid Batch asset with the BatchState Batch_Verified and a valid QA participant. When complete, this transaction results in the Batch asset having the BatchState Batch_Rejected.

```
173  transaction LocalBatchRejected extends BatchStateChange {
174    --> Batch BatchState
175    --> QA QA
176  }
```

*Figure 89 - The model file for the transaction LocalBatchRejected*

RegisterBatchInSoftwareD, shown in Figure 90, occurs after the batch is registered in software D by the warehouse. It requires a valid DoC asset with the DoCState Blank and a valid Warehouse participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Headline_Filled.

```
191  transaction RegisterBatchInSoftwareD extends DoCStateChange {
192    --> DoC DoCState
193    --> Warehouse Warehouse
194  }
```

*Figure 90 - The model file for the transaction RegisterBatchInSoftwareD*

WarehouseSectionFilled, seen in Figure 91, happens when the warehouse section of the DoC is filled by the warehouse. It requires a valid DoC asset with the DoCState DoC_Headline_Filled and a valid Warehouse participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Warehouse_Section_Filled.

```
199  transaction WarehouseSectionFilled extends DoCStateChange {
200    --> DoC DoCState
201    --> Warehouse Warehouse
202  }
```

*Figure 91 - The model file for the transaction WarehouseSectionFilled*

QualityReview, represented in Figure 92, occurs when the batch quality review is completed by the QA or QPPT. It requires a valid DoC asset with the DoCState DoC_Warehouse_Section_Filled and a valid QA participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Quality_Section_Filled.

```
207  transaction QualityReview extends DoCStateChange {
208    --> DoC DoCState
209    --> QA QA
210  }
```

*Figure 92 - The model file for the transaction QualityReview*

LocalRelease, demonstrated in Figure 93, happens when the batch is released locally by the QPPT. It requires a valid DoC asset with the DoCState DoC_Quality_Section_Filled and a valid QA participant with the Qposition QPPT. When complete, this transaction results in the DoC asset having the DoCState DoC_Nearly_Complete.

```
215  transaction LocalRelease extends DoCStateChange {
216    --> DoC DoCState
217    --> QA QA
218  }
```

*Figure 93 - The model file for the transaction LocalRelease*

CentralRelease, shown in Figure 94, occurs when the batch is released centrally by the QA. It requires a valid DoC asset with the DoCState DoC_Quality_Section_Filled and a valid QA participant. When complete, this transaction results in the DoC asset having the DoCState Central_Release_DoC.

```
223  transaction CentralRelease extends DoCStateChange {
224    --> DoC DoCState
225    --> QA QA
226  }
```

*Figure 94 - The model file for the transaction CentralRelease*

UploadToSoftwareC, seen in Figure 95, happens when the DoC is uploaded to Software C by the QA. It requires a valid DoC asset with the DoCState Central_Release_DoC and a valid QA participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Awaiting_Approval.

```
231  transaction UploadToSoftwareC extends DoCStateChange {
232    --> DoC DoCState
233    --> QA QA
234  }
```

*Figure 95 - The model file for the transaction UploadToSoftwareC*

DoCReview, represented in Figure 96, occurs when the DoC is reviewed by the QPEU. It requires a valid DoC asset with the DoCState DoC_Awaiting_Approval and a valid QPEU participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Approved.

```
239  transaction DoCReview extends DoCStateChange {
240    --> DoC DoCState
241    --> QPEU QPEU
242  }
```

*Figure 96 - The model file for the transaction DoCReview*

UpdateInSoftwareC, demonstrated in Figure 97, happens when the DoC is updated in Software C by the QPEU. It requires a valid DoC asset with the DoCState DoC_Approved and a valid QPEU

participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Nearly_Complete.

```
247  transaction UpdateInSoftwareC extends DoCStateChange {
248    --> DoC DoCState
249    --> QPEU QPEU
250  }
```

*Figure 97 - The model file for the transaction UpdateInSoftwareC*

CentralBatchUnredeemable, shown in Figure 98, occurs when the centrally released batch is classified as unredeemable by the QPEU. It requires a valid DoC asset with the DoCState DoC_Approved and a valid QPEU participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Rejected.

```
255  transaction CentralBatchUnredeemable extends DoCStateChange {
256    --> DoC DoCState
257    --> QPEU QPEU
258  }
```

*Figure 98 - The model file for the transaction CentralBatchUnredeemable*

BatchCorrupted, seen in Figure 99, happens when the batch is classified as corrupted by the QA or QPPT. It requires a valid DoC asset with the DoCState DoC_Quality_Section_Filled and a valid QA participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Rejected.

```
263  transaction BatchCorrupted extends DoCStateChange {
264    --> DoC DoCState
265    --> QA QA
266  }
```

*Figure 99 - The model file for the transaction BatchCorrupted*

LocalBatchUnredeemable, represented in Figure 100, happens when the locally released batch is classified as unredeemable by the QPPT. It requires a valid DoC asset with the DoCState DoC_Nearly_Complete and a valid QA participant with the Qposition QPPT. When complete, this transaction results in the DoC asset having the DoCState DoC_Rejected.

```
271  transaction LocalBatchUnredeemable extends DoCStateChange {
272    --> DoC DoCState
273    --> QA QA
274  }
```

*Figure 100 - The model file for the transaction LocalBatchUnredeemable*

UpdateStatusInSoftwareD, demonstrated in Figure 101, occurs when the batch status is updated in Software D by the QA or QPPT. It requires a valid DoC asset with the DoCState DoC_Nearly_Complete and a valid QA participant. When complete, this transaction results in the DoC asset having the DoCState DoC_Completed.

```
279  transaction UpdateStatusInSoftwareD extends DoCStateChange {
280    --> DoC DoCState
281    --> QA QA
282  }
```

*Figure 101 - The model file for the transaction UpdateStatusInSoftwareD*

#### *4.2.3.1 Transaction Remarks*

- ❖ The transactions change the state of each of the assets, the Batch and DoC, and were modeled after the triggers represented in the respective state machine diagrams;
- ❖ These can only be submitted by valid participants in the batch release process Blockchain business network;
- ❖ In order for transactions to work properly, a Script file (.js) with all the requirements must be created,
- ❖ The period of each stage of the batch release process can be accurately measured by checking the interval between timestamps of the corresponding transactions.

### 4.2.4 Model Testing

In order to validate the Blockchain Model file, a fictional sample population was created, consisting of:

- Four batches: 1111, 2222, 3333, 4444;
- Four DoC corresponding to each batch;
- One Warehouse participant: Manuel;
- Two QA participants, one a general QA and another a QPPT: Afonso, Maria;
- One QPEU participant: Joseph.

Each Batch asset was created by generating sample (but valid) corresponding attributes, as shown in Figure 102.

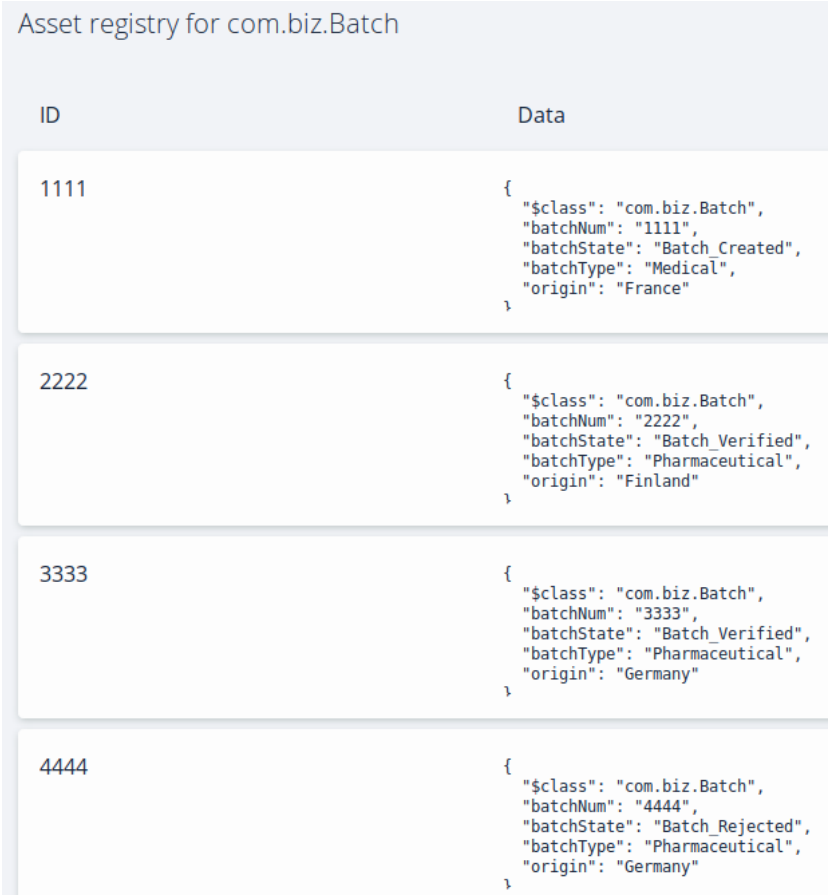


*Figure 102 - The sample Batch assets*

The DoC assets were created matching each of the Batch assets, as seen in Figure 103. The DoC asset also requires having a valid DoCState corresponding to its Batch BatchState (e.g. the Batch 1111 has the BatchState Batch_Created, therefore the only possible DoCState for DoC 1111 is Blank).


*Figure 103 - The sample DoC assets*

The warehouse participant was created by generating sample corresponding attributes, as shown in Figure 104.


*Figure 104 - The sample Warehouse participant*

The QA participants were created by generating sample corresponding attributes. Although, for the sake of diversity, each of the two QA participants has a different Qposition, one being a general QA, represented in Figure 105, and the other being a QPPT, represented in Figure 106.


*Figure 105 - The sample general QA participant*


*Figure 106 - The sample QPPT QA participant*

The participant QPEU was created by generating sample corresponding attributes, as seen in Figure 107.


*Figure 107 - The sample QPEU participant*

Adding the assets and participants created log entries of the type *AddAsset,* as shown in Figure 108. Each entry also has the date and time it was created and the network participant who created the transaction.

| Date, Time | Entry Type | Participant |
|---|---|---|
| 2018-10-29, 17:31:00 | AddAsset | admin (NetworkAdmin) |
| 2018-10-29, 17:29:50 | AddAsset | admin (NetworkAdmin) |
| 2018-10-29, 17:28:23 | AddAsset | admin (NetworkAdmin) |
| 2018-10-29, 17:25:55 | AddAsset | admin (NetworkAdmin) |
| 2018-10-29, 17:24:14 | AddAsset | admin (NetworkAdmin) |
| 2018-10-29, 17:22:36 | AddAsset | admin (NetworkAdmin) |
| 2018-10-29, 17:21:50 | AddAsset | admin (NetworkAdmin) |

*Figure 108 - The resulting transaction log after creating the sample population*

Each of the *AddAsset* log entries states what was added, where it was added, shown in *targetRegistry* (in Figure 109‘s example, in the Batch assets), has its unique digital footprint, shown in *transactionId*, and a timestamp.

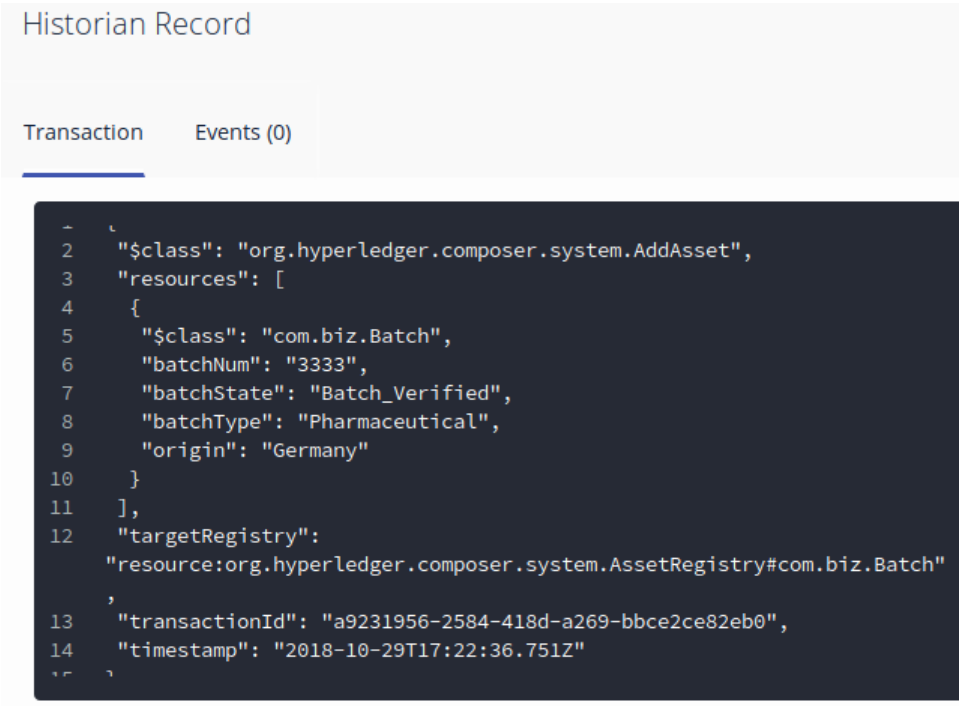


*Figure 109 - The creation entry of the sample asset Batch 3333*

#### *4.2.4.1 Model Testing Remarks*

- Testing shows that the model file asset and participant creation work as planned, because the sample population was successfully created as intended;
- There is no way of verifying the transaction submission as this would require the Script file (.js) to be implemented.

## 4.3 Discussion & Future Work

The SysML model showed the several delays points of batch release process, including some unexpected by the Company, like the delay in the Pre-wholesaler. To improve the accuracy of this model, more details about the process are required, like the number of deviations found. Data like this will allow the drawing of a graph showing the behavior over time for each of the Variable blocks. These graphs help provide a better evaluation of the current situation of the Company. This is done with the creation of performance KPI's using the Variables data as inputs. For example, using the variables *Deviations Found* and *Deviations Found by QPEU* as inputs, a KPI for the effectiveness of the QPEU can be created, which provides the percentage of defective batches found by this actor. These KPI's allow the Company to act according to the behavior observed, to improve the system performance.

To provide accurate data for input to the Variables, the Blockchain model file could be implemented alongside the current Company software. This would require the other complementary Blockchain business architecture files in order for it to work properly. The Script file (.js), which contains the system requirements that allow the model file to work, and the Access Control file (.acl) which dictates the access permission to the system, preventing unauthorized personnel to alter or access the information contained within the files. When implemented, the complete Blockchain application provides the Company with precise real-time data obtained from the other software currently in use.

The Blockchain model file testing shows promising results, as each of the elements of the population was created the way it was pretended, in conformity with the SysML model of the batch release process. If for any valid reason, new information or parameters are required to add to the model, this is possible by erasing the former component, altering the model file, and creating the new component with the updated information. The Blockchain will record all these changes however, including the erasing or creation of a new asset or altering the model file.

# 5. Conclusion

Developing the Blockchain business architecture foundation for the Company's batch release process is the first step for coping with the delay points within the system. This architecture could only be built by having deep understanding of how the process works, which was accomplished with the creation of a SysML model of the system. This model was build using the information and data provided by the Company, during the several meetings held.

The SysML model showed multiple delay points detected in the system, including some unexpected by the Company (such as a delay in the Pre-Wholesaler's Warehouse), and how long these delays were. The model also worked as the framework for creating the Model file (.cto) of the Blockchain business architecture. Due to the several common elements shared between the two software, the creation of the Model file in Hyperledger Composer Playground was very smooth.

To complete the implementation of the Blockchain business architecture, two other files are required, the Script (.js) and the Access Control file (.acl). The first describes what are the system requirements for the model file, while the latter states what entities can access or alter the data within the system.

The implementation of a Blockchain platform allows the Company to evaluate its current situation. Using the information obtained from the platform (e.g. the time each process stage takes), the Company can create performance KPI's (e.g. the time the warehouse takes to perform its tasks) that allow it to take action according to the performance observed (e.g. replacing the Pre-Wholesaler). It will also allow the Company to verify if redundant steps (e.g. DoC review by the QPEU) are required in the batch release process.

Acting on the source of the delays found in the system will result in reducing the total duration of the batch release process and, consequently, diminish the Company's inventory costs currently estimated at 5M€. If this value is proportional to the duration of the batch release process, making it go from nine to five days (40% reduction) corresponds to over 2M€ savings for the potential value of this project.

# 6. References


[1] S. C. Spangelo *et al.*, “Model based systems engineering (MBSE) applied to Radio Aurora Explorer (RAX) CubeSat mission operational scenarios,” *IEEE Aerosp. Conf. Proc.*, 2013.

[2] C. Shamieh, *Systems Engineering For Dummies, ®IBM Limited Edition*. Wiley Publishing, Inc., 2011.

[3] D. Thomas, “What is SysML? The Systems Modelling Language Explained,” *D Thomas Software*, 2015. [Online]. Available: http://dthomas-software.co.uk/resources/frequently-asked-questions/what-is-sysml-2/. [Accessed: 02-May-2018].

[4] S. J. I. Herzig, R. Karban, G. Trancho, F. G. Dekens, N. Jankevičius, and M. Troy, “Analyzing the Operational Behavior of the Alignment and Phasing System of the Thirty Meter Telescope using SysML.”

[5] R. Karban, “Towards Model Re-usability for the development of telescope control systems,” 2009.

[6] H. Graves, S. Guest, J. Vermette, and Y. Bijan, “Air Vehicle Model-Based Design and Simulation Pilot,” 2009.

[7] S. Corns and C. Gibson, “A model-based reference architecture for medical device development,” *22nd Annu. Int. Symp. Int. Counc. Syst. Eng. INCOSE 2012 8th Bienn. Eur. Syst. Eng. Conf. 2012, EuSEC 2012*, vol. 4, pp. 2731–2740, 2012.

[8] P. Pearce and S. Friedenthal, “A Practical Approach For Modelling Submarine Subsystem Architecture In SysML,” *Submar. Inst. Aust. Sci. Technol. Eng. Conf.*, pp. 347–360, 2013.

[9] P. Pearce and M. Hause, “Model-Based Submarine Design,” no. Mitchell 2010, 2012.

[10] S. W. Mitchell, “Model-Based System Development for Managing the Evolution of a Common Submarine Combat System,” no. May, pp. 1–30, 2010.

[11] R. Karban, M. Zamparelli, B. Bauvir, B. Koehler, L. Noethe, and A. Balestra, “Exploring Model Based Engineering for Large Telescopes - Getting started with descriptive models,” *Proc. SPIE*, vol. 7017, no. 1I, 2008.

[12] D. Richards, A. Stuart, and M. Hause, “Testing Solutions through SysML / UML,” *INCOSE Int. Symp.*, vol. 19, no. 1, pp. 760–774, 2009.

[13] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML The Systems Modeling Language, Third Edition*. Elsevier, Inc, 2015.

[14] T. Weilkiens, *Systems Engineering with SysML/UML Modeling, Analysis, Design*. Morgan Kaufmann OMG Press, 2006.

[15] D. M. Buede, *The Engineering Design of Systems: Models and Methods, 2nd Edition*. Wiley Publishing, Inc., 2009.

[16] Center for Strategic and International Studies, “SM-65 Atlas | Missile Threat,” 2017. [Online]. Available: https://missilethreat.csis.org/missile/atlas/. [Accessed: 24-Jun-2018].

[17] P. Baracos, *Grafcet Step-by-Step*. Famic Automation, 1992.

[18] M. Higginson, J.-T. Lorenz, B. Münstermann, and P. B. Olesen, “The promise of blockchain,” *McKinsey Q.*, 2017.

[19] K. Nash, “IBM Pushes Blockchain into the Supply Chain,” *Wall Street Journal*, 2016.

[20] K. Wang and A. Safavi, “Blockchain is empowering the future of insurance,” *TechCrunch*, 2016.

[21] I. Heap and D. Tapscott, “Blockchain Could Be Music’s Next Disruptor,” *Fortune*, 2016.

[22] P. Evans, L. Aré, P. Forth, N. Harlé, and M. Portincaso, “Thinking Outside The Blocks: A Strategic Perspective on Blockchain and Digital Tokens,” 2016.

[23] “Lisk Academy » Learn about Blockchain Technology.” [Online]. Available: https://lisk.io/academy. [Accessed: 12-Nov-2018].

[24] H. Min, “Blockchain technology for enhancing supply chain resilience,” *Bus. Horiz.*, 2018.

[25] “Blockchains: The great chain of being sure about things,” *The Economist*, 31-Oct-2015.

[26] K. Nash, “Blockchain: Catalyst for Massive Change Across Industries,” *Wall Street Journal*, 2016.

[27] “Ujo.” [Online]. Available: https://ujomusic.com/. [Accessed: 21-Sep-2018].

[28] A. F. Hussein, N. ArunKumar, G. Ramirez-Gonzalez, E. Abdulhay, J. M. R. S. Tavares, and V. H. C. de Albuquerque, “A medical records managing and securing

blockchain based system supported by a Genetic Algorithm and Discrete Wavelet Transform," *Cogn. Syst. Res.*, vol. 52, pp. 1–11, 2018.

[29] K. A. Koshechkin, G. S. Klimenko, I. V. Ryabkov, and P. B. Kozhin, "Scope for the Application of Blockchain in the Public Healthcare of the Russian Federation," *Procedia Comput. Sci.*, vol. 126, pp. 1323–1328, 2018.

[30] C. Sullivan and E. Burger, "E-residency and blockchain," *Comput. Law Secur. Rev.*, vol. 33, no. 4, pp. 470–481, 2017.

[31] "IBM Blockchain Platform: Develop," *Github*. [Online]. Available: https://ibm-blockchain.github.io/develop/business-network/businessnetworkdefinition. [Accessed: 20-Sep-2018].

# 7. Annex A: How to install a local version of the Hyperledger Playground Composer

While the online Playground runs the business network in browser local memory, requiring internet connection, the local Playground is deployed in Hyperledger Fabric, the infrastructure behind Hyperledger Composer.

The online version can be access at: *https://composer-playground.mybluemix.net/*

Local Version Required components:

- Operating Systems: Ubuntu Linux 16.04 LTS (64-bit) or Mac OS 10.12;
- At least 4Gb of memory;
- Docker Engine 17.03 or greater;
- Docker Composer 1.8 or greater.

Recommended Components:

- A Virtual Box, in case neither of the required but a Windows OS is currently available for the user;
- An image for one of the required OS, although Ubuntu is better, since it's an open source software and therefore easier to obtain.

To install Docker and Docker compose in Ubuntu 16.04 LTS (64-bit) use the console commands in order:

- install docker
*$ sudo apt install docker.io*

- install docker compose
*$ sudo apt install docker-compose*

- create a user group
*$ sudo usermod -a -G docker $USER*

If Hyperledger Fabric or the local version of Hyperledger Composer Playground have been used in the machine, and it is desired to start everything anew, the following commands should be used. The user should be careful if there any other docker images on the machine.

*docker ps -aq | xargs docker rm -f*

*docker images -aq | xargs docker rmi -f*

To create the containers and install the Playground locally use:
*curl -sSL https://hyperledger.github.io/composer/install-hlfv1.sh | bash*

Or
*docker run --name composer-playground --publish 8080:8080 hyperledger/composer-playground*

When installed use the following command to restart the Playground:
*./composer.sh*

A playground is now running, and it can be accessed in a web browser by following the address: *http://localhost:8080/login*

# 8. Annex B: Meetings

- Meeting 21/06/2017

The reasoning for this meeting was to decide the project I should do in collaboration with the Company. Currently, one of the main concerns of the Company is the high amount of time spent in the release process of the batches. Since the value of the products in stock waiting to be released is about 5M€, we agreed that would be the main focus of the project.

- Meeting 3/08/2017

During this meeting, the confidentiality agreement terms were validated, and the project subject was further discussed and detailed. The Company also gave us access to the documentation about their current Batch Release process.

- Meeting 9/11/2017

The confidentiality agreement was signed and a more in-depth explanation about the Batch Release process was given to us. A Pre-Wholesaler is responsible for storing the batches while they wait for release, and those batches include products from two branch Companies of the main Corporation, which share the release process. A visit to the Pre-Wholesaler's warehouse was booked to better understand the steps of the release process on which this entity is involved.

- Meeting 20/12/2017

In this meeting, we presented the modelling tool that's being used to describe the system and some of the progress so far. The Company explained to us more details about the process we observed at the warehouse. The Pre-Wholesaler has one day at maximum, established in a contract, to register in a software the batches after their arrival at the warehouse. Buy orders are placed by the demand management and need to consider the current lead time of releasing the products. This is about nine days for the centrally released batches (products made in the EU), five days for locally released batches (products produced locally) and about a month for batches produced outside the EU. These high lead times mean a high inventory cost due to the storage of the batches while these wait to be released. We agreed that the scope of our project would not include products not made in the EU. The Company agreed to draw us a diagram to better explain the release process.

CONTROLO DE RECEPÇÃO E DE LIBERTAÇÃO DE LOTE

Recepção informática:
Organização: PT01
Grupo compras: [redacted]
Nº Pedido:

Recepção física:
Denominação: [redacted]
Telefone:
Shipping advice:

Material:732998
Denominação:
Nº Registo:

Nº Lote:

Quantidade:

---

INFORMAÇÃO DE ARMAZÉM

Armazém: 1001 Initial GR PH

Nota recepção: 000000893787

| Contentores | Resultados | Observações |
|---|---|---|
| Aspecto | _____ | _____ |
| Rotulagem | _____ | _____ |
| Nº unidades/cx grupagem 1 | _____ Data | _____ |

Verificado por: _____ Data _____

---

INFORMAÇÃO DE GARANTIA DE QUALIDADE

Amostra _____ Aspecto exterior _____ Aspecto interior_____

| Artworks | Referência | Nº lote | Data validade |
|---|---|---|---|
| Folheto Informativo | _____ | | |
| Cartonagem | _____ | _____ | _____ |
| Alumínio / rótulo | _____ | _____ | _____ |
| Outros | _____ | _____ | _____ |
| | _____ | _____ | _____ |
| | _____ | _____ | _____ |

Nº registo PT_____ Nº registo UE_____ PVP _____

Observações: _____

Verificado por: _____ Data _____

---

CoA fabricante / laboratório UE _____
MT fabricante / laboratório UE _____

Condições transporte _____
Reclamação técnica _____ Desvio _____

Observações _____

Verificado por _____ Data _____

---

Este lote foi fabricado, analisado e libertado de acordo com a Autorização de Introdução no Mercado e as Boas Práticas de Fabrico em vigor na UE.

Estatuto de lote _____ Assinatura _____ Data _____

Alteração do estatuto de lote efectuada em SAP por_____ Data_____

*Figure 110 - The DoC currently in use*

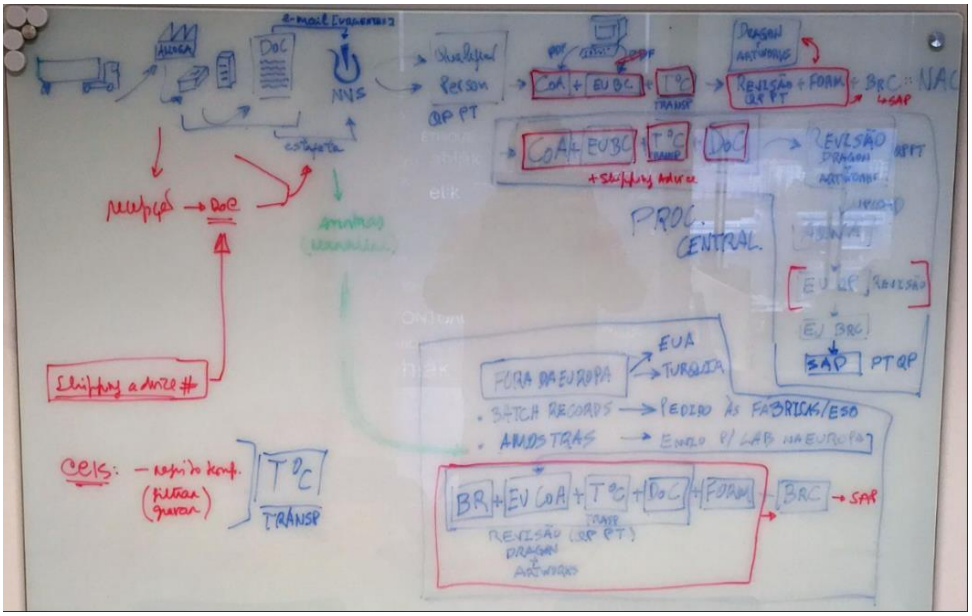

*Figure 111 - The Drawing of the Release Process*

- Meeting 22/01/2018

Our doubts about the drawing the Company sent us were clarified. The local release is made by a QP which uses three software to do it: CCIS, Dragon and SAP; while the central release also requires a EUQP and uses a forth software named AQWA. The products, and the artworks about them, are in constant update and if they do not match the latest version, they cannot be released. Depending on the batch, these can be updated locally, centrally or both. Rejected batches are destroyed. It was also explained to us how the Shipping Advice and the DoC relate to each other.

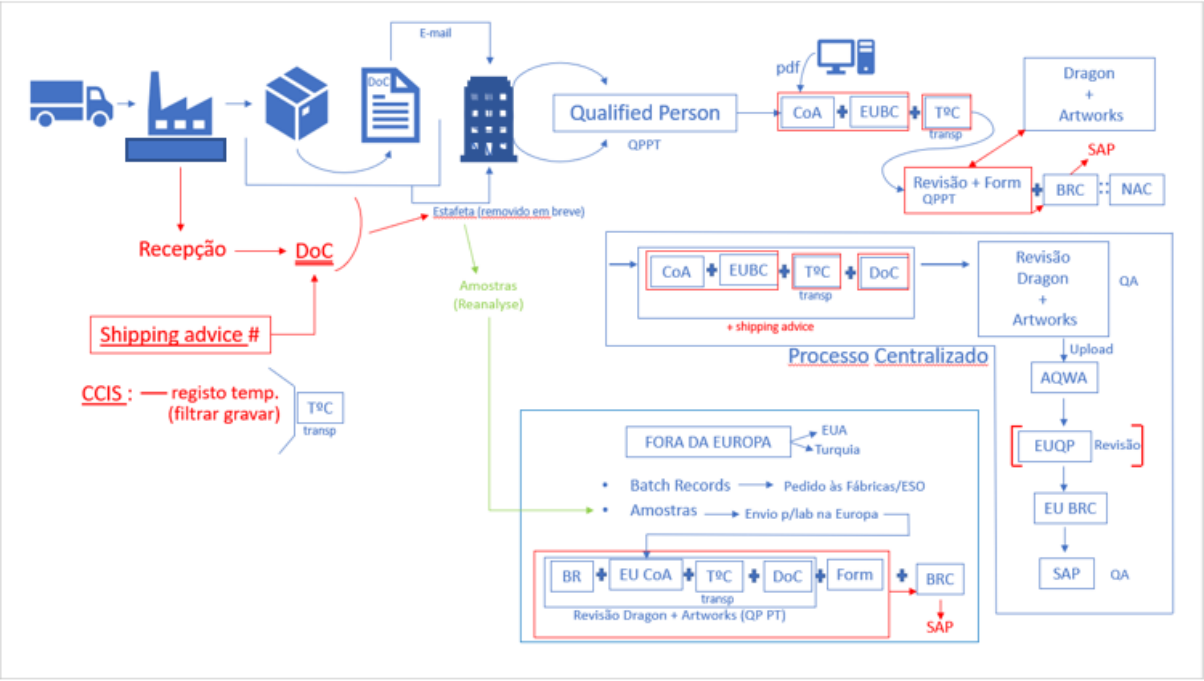


*Figure 112 - The clear and updated drawing of the Release Process*

- Meeting 01/03/2018

In this meeting, it was explained that 60% of the batches are released centrally. The difference between the several types of DoC's is just a matter of which one is easiest to fill and the DoC currently in use is being subject to change at the current time. The Company agreed on sending the time logs of the several different software platforms related to the batches of a certain time period. It was revealed that the Pre-Wholesaler now uses a software platform that replaces the courier which transports the batch documentation to the Company. A recent alteration in the central release process states that the QP is replaced by the QA for doing certain tasks in the process.

- Meeting 09/04/2018

A member of the IT department was present in this meeting to help us extract the logs from the software. We concluded that the relevant logs of interest were from SAP, AQWA and the Pre-Wholesaler's software. The first two, we could extract the data related to January's batches, while the third, only the Pre-Wholesaler has access to, although we can ask for the logs if we need them in a future stage of the project. We talked about a fourth software, the CCIS, which the logs could be of interest, although the relevant data cannot be extracted and needs to be manually obtained. If needed, this information can be provided in a later stage of the project.

| Material | Material Description | | | | Plnt | Name 1 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reference | SLoc | MvT | S | Mat. Doc. | Item | Pstng Date | Quantity in UnE | EUn | Amount in LC | Batch | Time | PO | User name |
| 135730 | STARLIX 120MG 84CP | | | | | | PT01 | | | | | | |
| | 1001 | 101 | | 5002963947 | 1 | 21.02.2018 | 1.512 | PC | 19.565,28 | TN054 | 13:50:29 | 4700098437 | WF-BATCH |
| | 1001 | 101 | | 5002963948 | 1 | 21.02.2018 | 2.184 | PC | 28.260,96 | TN054 | 13:50:30 | 4700098563 | WF-BATCH |
| * Total | | | | | | | | | | | | | |
| | | | | | | | 3.696 | PC | 47.826,24 | | | | |

*Figure 113 - An example of a SAP log*

- Meeting 23/04/2018

During this meeting, it was explained that in the software AQWA, the *Opened On* entry is created by the QA locally, meaning we can conclude how many days are spent in the Central QA. The entry *Transport Eval. Completed On* only registers the last recorded date, meaning, inside the central release process there might be a need for several reevaluations and the system only saves the last time an evaluation was completed. The Central QA usually takes three days to perform their tasks relevant to the release process, this is due to the high volume of work this entity needs to do. The DoC's are only printed after the batch is registered in the SAP software, therefore, the shipments might physically arrive earlier than their system registry. To check how much time is actually spent between the physical and system reception at the warehouse, the data logs from the CCIS system can be used, and for this reason the Company supplied the CCIS logs relevant to the batches corresponding to January.

- Visit to the Pre-Wholesaler warehouse 14/11/2017

During this visit we observed the batch reception process at the warehouse. It went as follows:

- The truck arrives, and its unloading is manual.

- Then, the manual exterior verification of the batches begins. Any deviation found (besides the ones noted by the producer beforehand) is forwarded to the account manager.
    - When verifying the price tag of the batch, if it matches with the price table, it receives a green label with “OK” on it and the information system decides where the batch is stored. If it does not correspond to the price table, it receives a red label and the batch undergoes a labelling process where its price is updated to match the new one.
    - From each verified batch, a sample is collected, so it can later be released by the Company. This sample is delivered to the account manager for a detailed verification.
- The batch storing and dispensing is made by three shifts of workers (a morning, a lunch and an afternoon shift), who receive the picking information, meaning, what batches to move and where to move them, from an automated dispenser earpiece.
    - In each of the shifts, only one type of action is made, either storing or collecting batches from the shelves. This is to prevent workers from colliding with each other. The earpiece is assigned to each individual worker, reacting only to the voice of its respective user. To avoid batch positioning errors, the earpiece only pronounces the last four digits of the batch, which are a unique combination.
    - To prevent overloading the workers, they are assigned to forty lines of batches each. If the shipment is very large, then more workers will be assigned to it.
    - When dispensing the batches to the Wholesaler, the cargo is first checked before loading the truck. The outgoing truck driver confirms the shipment by signing a dispensing form.
- The reception in the informatic system is made the account manager. This reception is what gives the directions to the dispenser earpiece to where to store the batch and provides a list of the products received in the warehouse.
    - The account manager starts by reverifying the sample’s documentation and taking copies for sending to the Company. Within these documents is the transport identification belonging to the sample and the temperature registry.
    - If a product is of urgent need, the account manager makes its reception first.
    - Considering the buy orders (received seven days earlier) the account manager checks-in the cargo that arrived at the warehouse.
        - During this check-in, the deviations found during the manual verification are here registered.
        - Each buy order is specific for each type of product and can have several batches associated with it.
        - The samples and units with deviations are only checked-in when the quality department of the Company approves it.
        - Samples are usually destroyed after being inspected, with exception of products with a high production cost.
    - Green price labels with “OK” on it are reverified.
        - Some products are free from this task.

        - The price between batches of the same product might be different, since one them might be older than the other, and a price change might have happened in the between the two production dates.
        - With the Company it is uncommon sudden changes to the price list.
    - About the temperature registry:
        - It's date and time are registered when the batch enters the warehouse.
        - There might be several temperature registries for the same transport, due to some batches being subject to different temperatures from others.
        - The temperature registry is more information for the quality guarantee of the Company.
    - About verifying the sample:
        - This is made by an account manager, where it is checked if the packaging of the product meets the standards of the Company.
        - After receiving the registry of verification, the Company can either destroy or forward the sample to another place.

# 9. Annex C: SysML Diagrams




*Figure 114 - Block Definition Diagram*

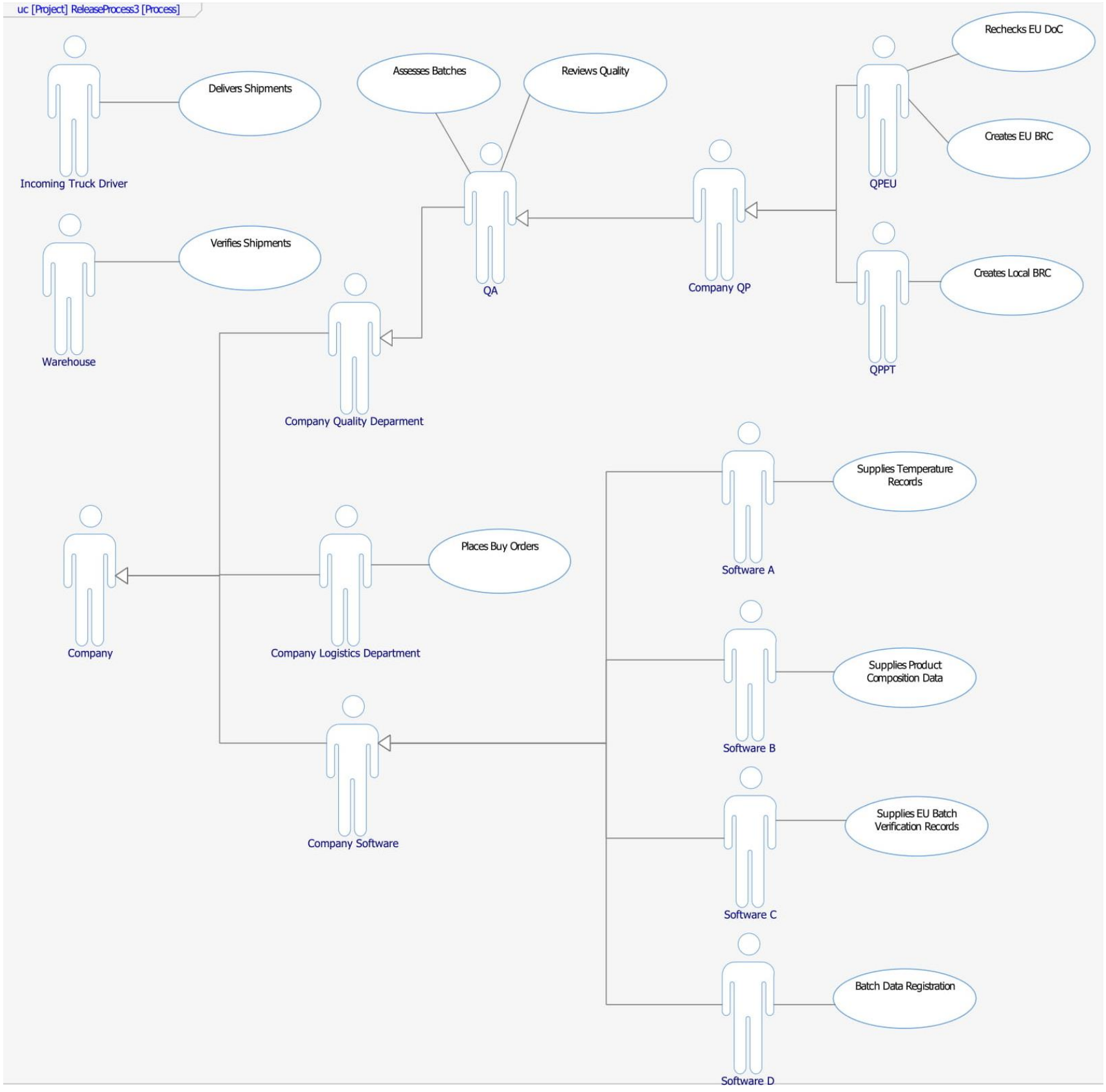


*Figure 115 - Use Case Diagram*

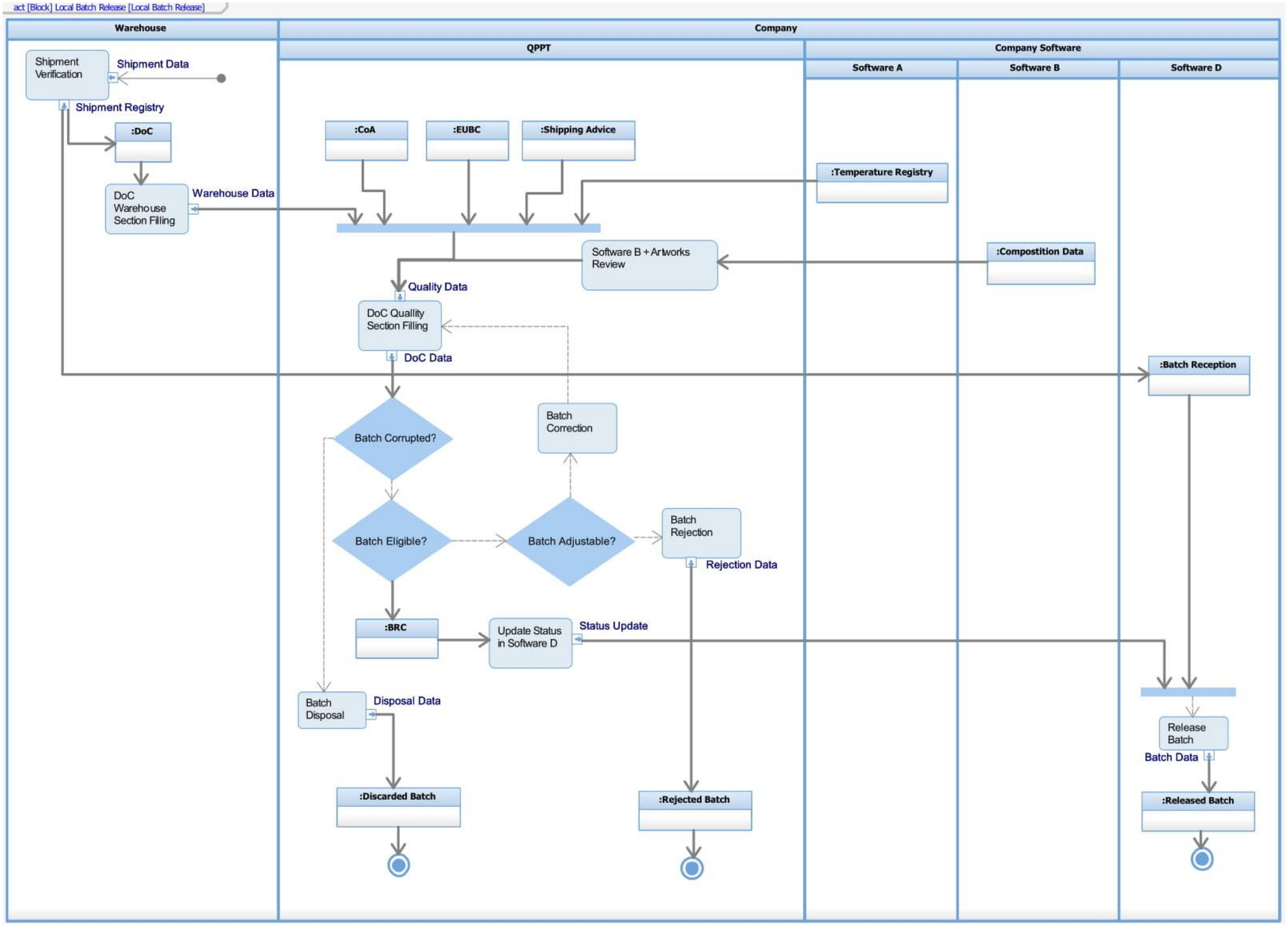


*Figure 116 - Local Release Activity Diagram*

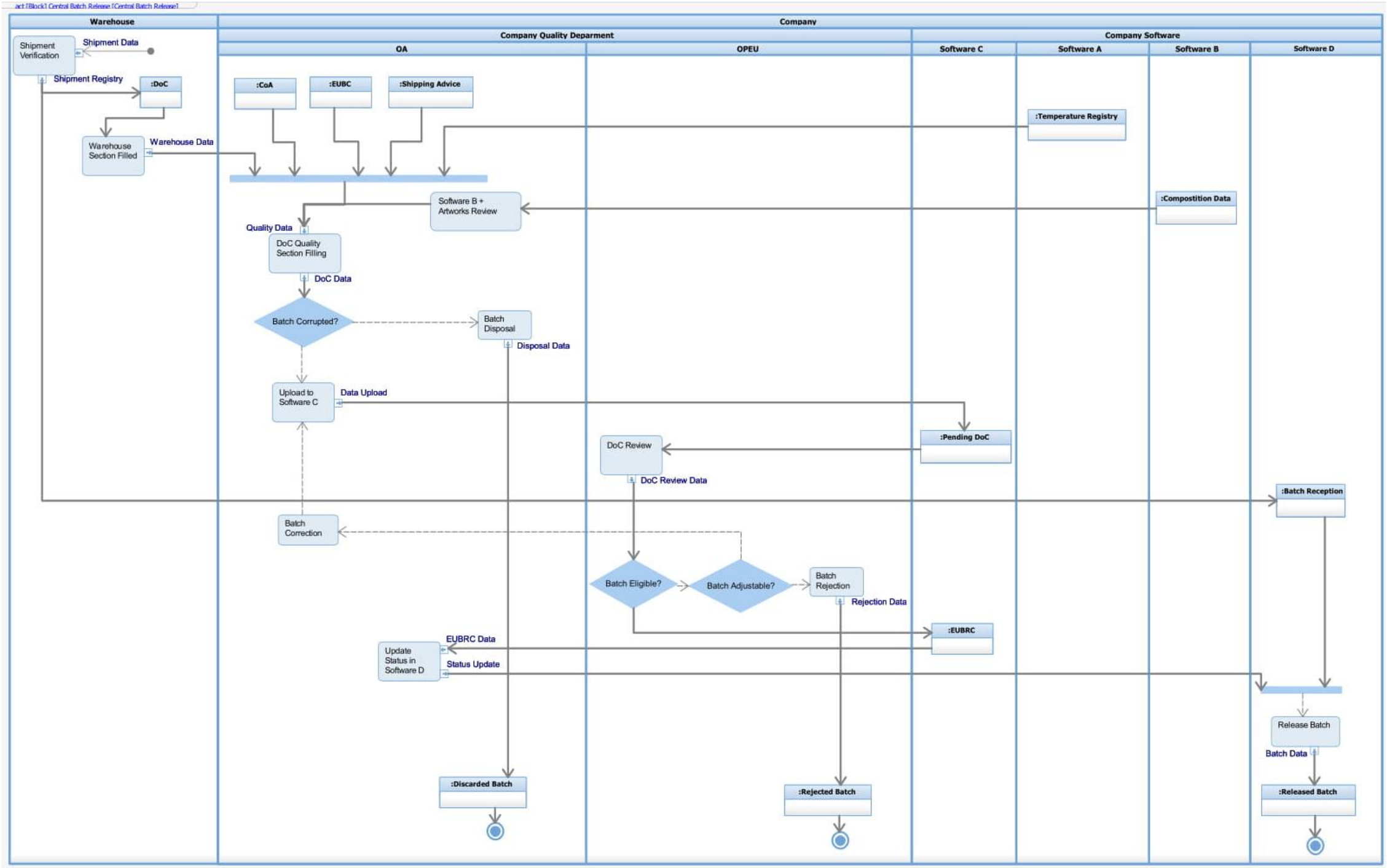


*Figure 117 - Central Release Activity Diagram*

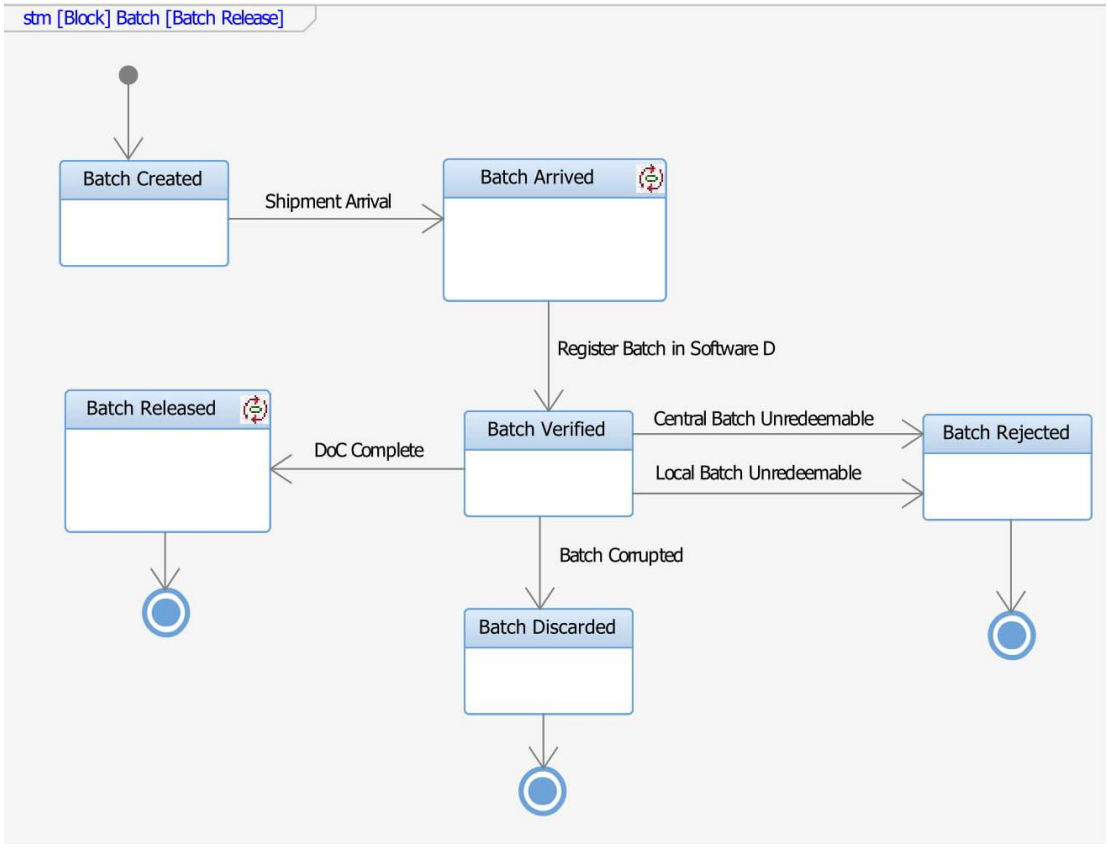


*Figure 118 - Batch State Machine Diagram*

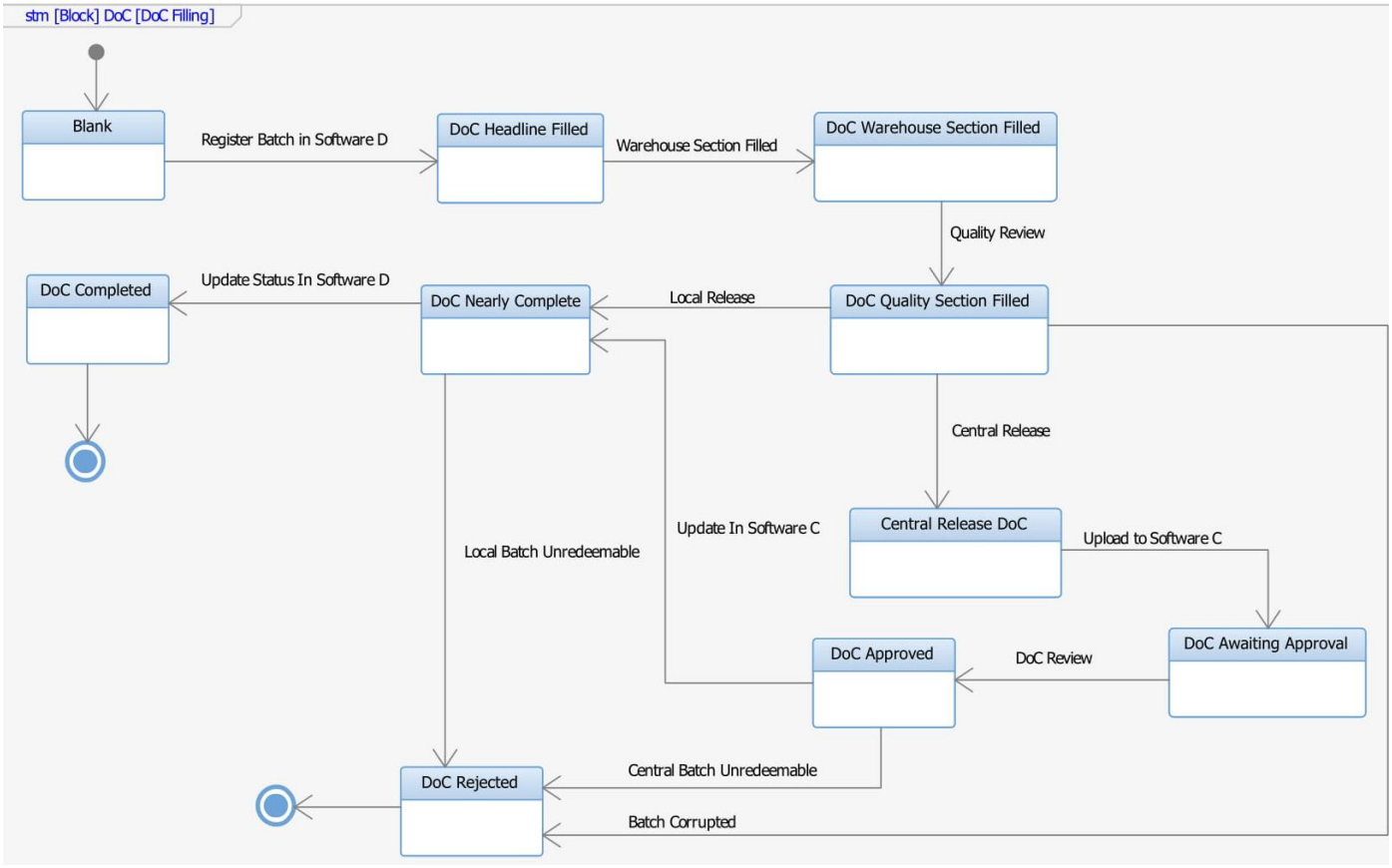


*Figure 119 - DoC State Machine Diagram*

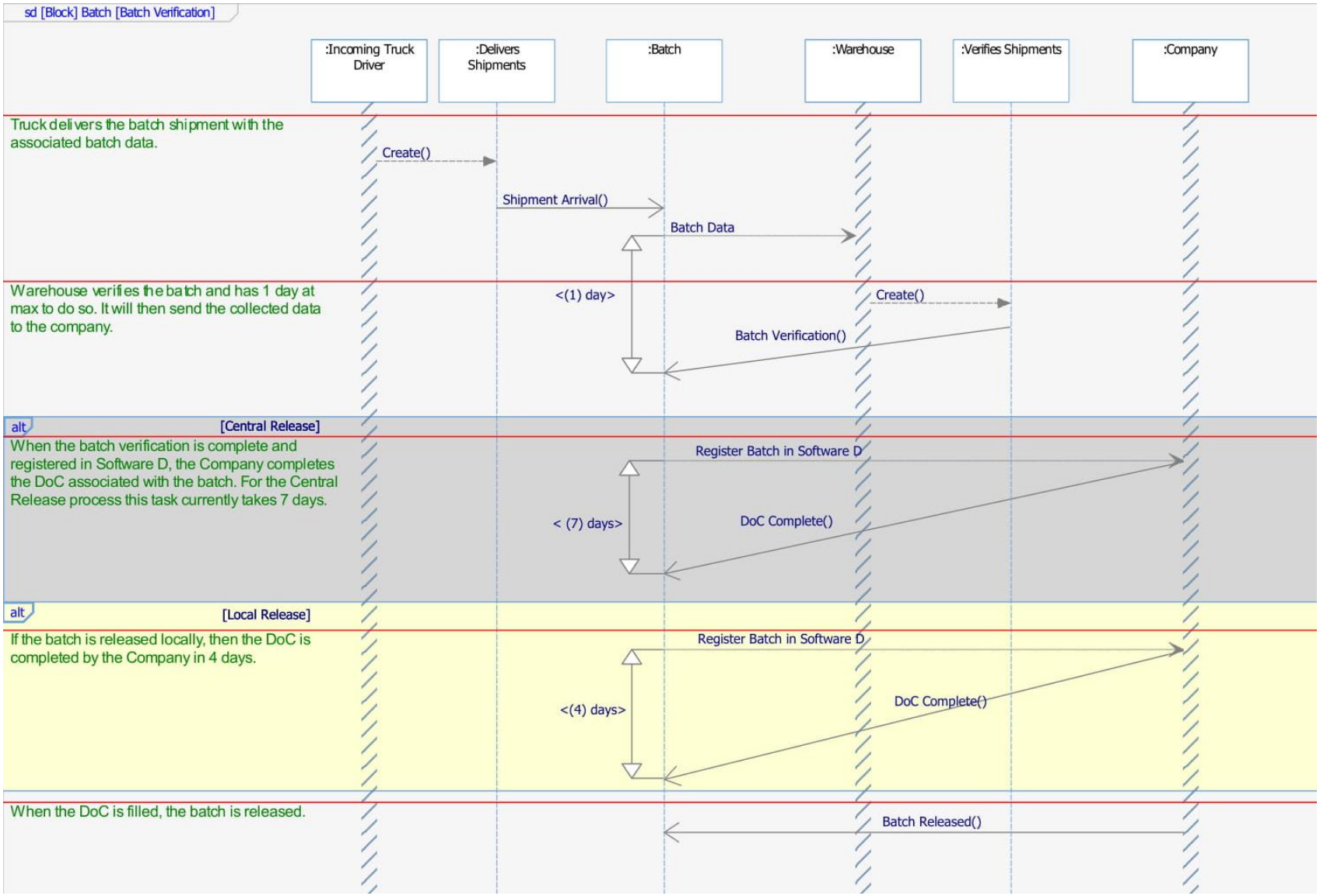


*Figure 120 - Batch Sequence Diagram*

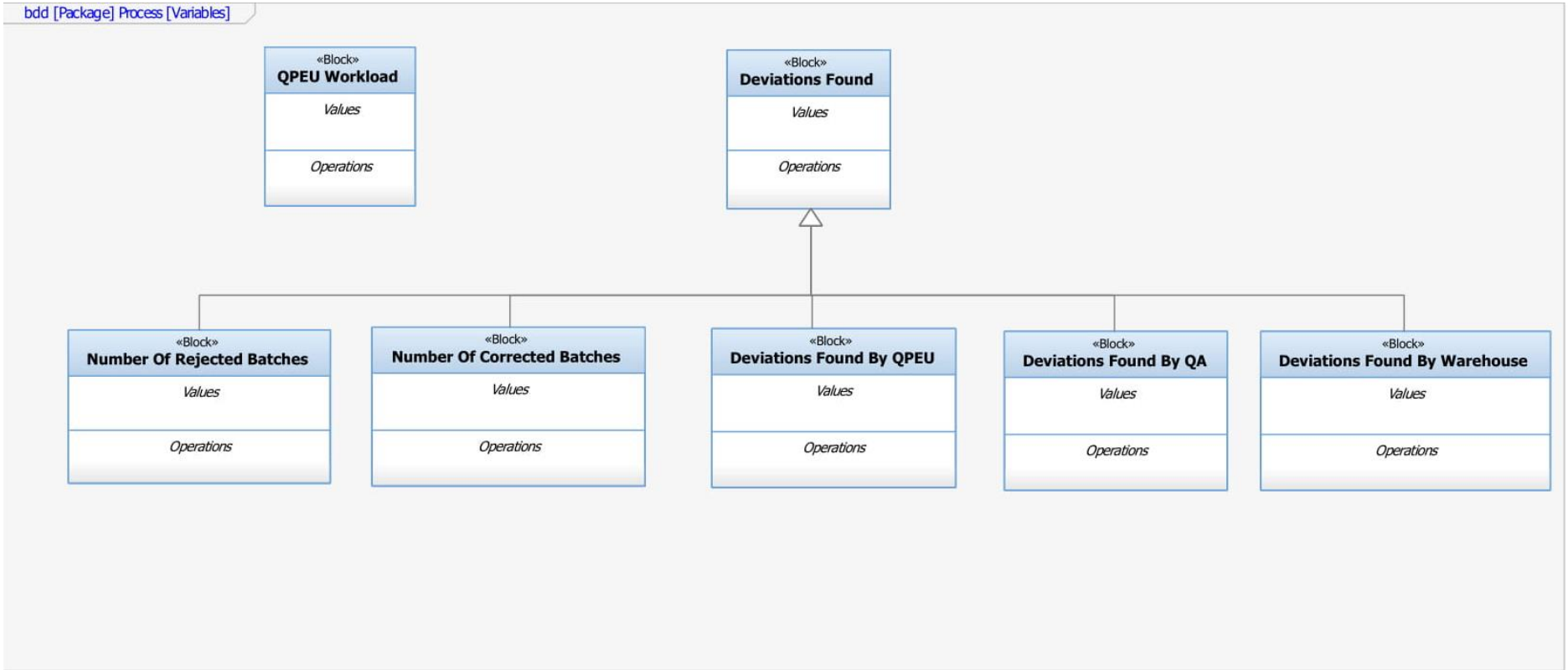


*Figure 121 - Block Definition Diagram of the Variables*

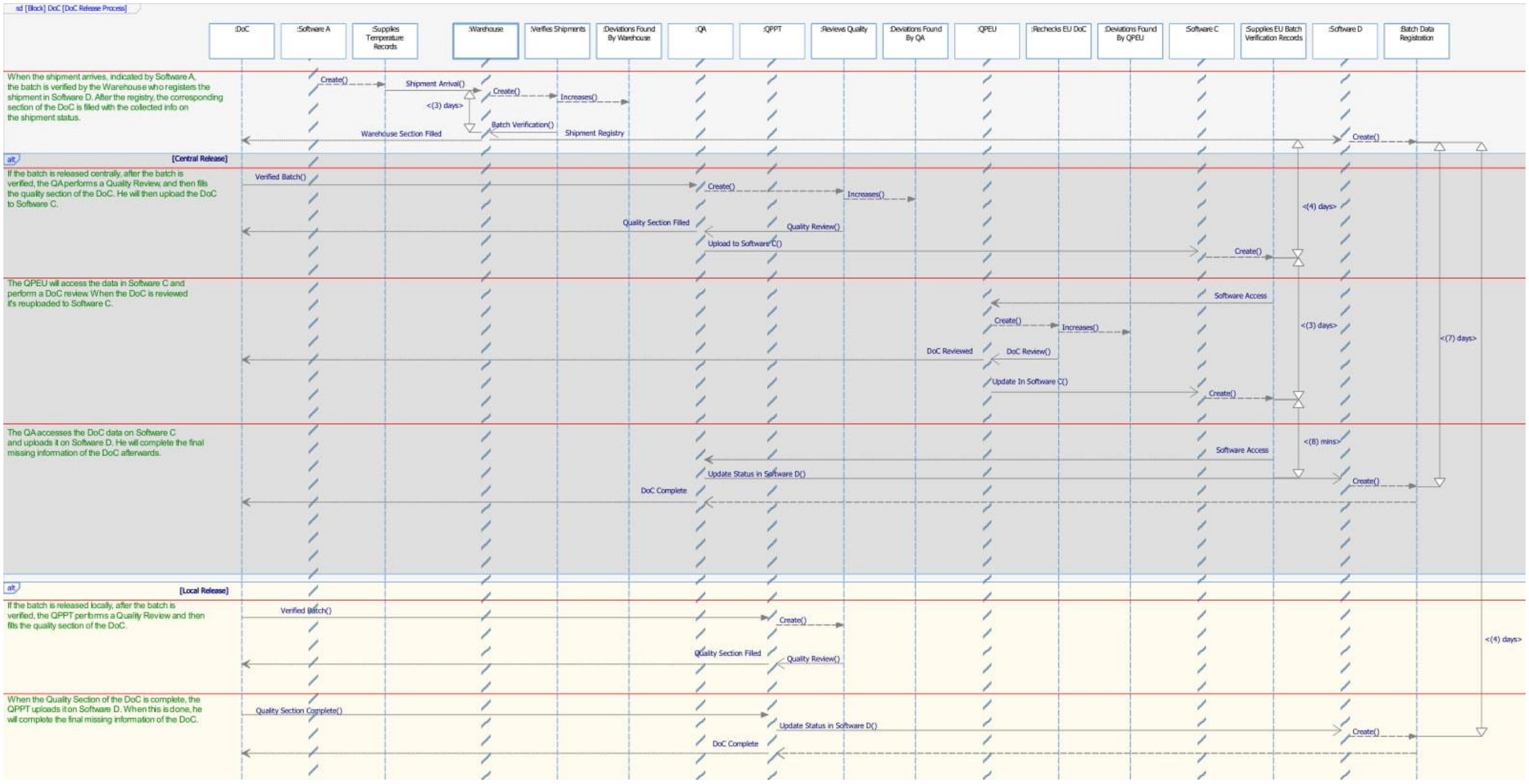


*Figure 122 - DoC Sequence Diagram*