Knowledge graph construction of User Stories components based on NLP and Neo4j

Ayodeji Ladeinde
BSc.(Mathematical Science) Unaab, PGD.(Finance) Unilag

Abstract—User stories are brief, basic descriptions of a system feature told from the point of view of the person who wants the new capability, which is usually a system user or customer/client. These stories are documented using simple natural language to help facilitate easy comprehension by different stakeholders working on creating the new system or adding new features to an existing one. However, depending on the system's complexity being designed or amended, these documented stories can be vast and uneasy for stakeholders to grasp what they relate to quickly or derive some insights just by reading them. This paper proposes the construction of the knowledge graph of users stories based on NLP and Neo4j. We extracted entities and relationships in these stories using a natural language processing algorithm, then constructed a knowledge-based using a graph database. The visualization obtained can be queried for any entity in the story, returning insights to enable stakeholders to make informed decisions. Helping to manage solutions with a substantial user requirement base, interactively and intuitively, aiding analysts to grasp the relationships between components and query different aspects of the requirements. This paper showed that graphs databases are suitable for handling many complexes, dynamic, interactive, low structured data like our user stories, whose components have been extracted using an NLP algorithm.

Index Terms—Knowledge Graph, User Stories, Neo4j, NLP, entity extraction.

1 Introduction

Software solutions are created by systematic scientific and technological methods defined as software engineering as stated in IEEE ISO document on systems and software engineering–vocabulary [1]. To embark upon real-world projects, stakeholders require an agreement on how the solution would work. This agreement is either something a stakeholder wants the software to do or a textual representation of their requirements as opined by Bourque et al. in [2]. These requirements aid the design and development of projects enabling a proper flow of information between different stakeholders. This help informs effective planning and assessment of the work [3] expected to be delivered. Again, as needs change, these requirements also vary, impacting the costs of resources invested in the requirements engineering process [4]. Thus, the need for a more adaptable approach to address the frequent changes of user needs that impacts every software engineering process. This need informed the birth of the "Manifesto for Agile Software Development" [5]. The manifesto is a collection of values and principles to guide the collaborative creation of working software in changing environments [4].

Nowadays, the business world is characterized by complexity, and market requirements are changing more rapidly. Solution providers are faced with limited time to get their products to market while ensuring they remained innovative and capture their customers' hearts [6].

Agile software development (ASD) methods promises benefits, such as on-time delivery and customer satisfaction [7], delivering business value in short iterations. Although, about 94 per cent of software professionals worldwide have successfully implemented agile approaches such as Scrum, Kanban, and XP [8]. However, according to Schon et al. in 2007 [9], these approaches fall short of identifying the right sort of product that meets user demands. It is recognized that to fill the gap in developing products with a good user experience (UX), a hybrid of ASD and Human-Centered Design (referred to as User-Centered Design by [10]) approaches should be applied. Our focus is not on the deficiency of the approaches, but on the visualization of the entities, objects, and relationships contained in the list of documented user stories captured during the ASD process using a knowledge graph.

A user story or requirement is a brief description of what a piece of software is meant to do, given by the person who wants the new feature [4] written in human natural language. Today through the use of a range of theory-motivated computational practises, the automatic analysis and representation of human language have continued to advance. We now have search engines like Google and similar technologies that process natural human language more petite than a second with reasonably high accuracy. Due to its simplicity, Natural language (NL) has impacted how practitioners document user, systems and software requirements. Research has shown that many practitioners who adopt the ASD framework document their user requirements artefacts in user stories using natural language notations. These requirements are written in story form as short pieces of texts in a semi-structured format such as "As a ¡type of user¿, I want ¡goal¿, [so that ¡some reason¿]".

As the use of Natural language processing continue in an upward direction, automatic creation of contextual models from series of documented user stories has become increasingly possible, contributing to rapid stakeholders'

- *A. Ladeinde is a current Masters of Data Science Student with the Faculty of Science, Engineering and Built Environment at Deakin University, Burwood, Melbourne, VIC, 3125.*
  E-mail: aladeinde@deakin.edu.au

understanding of user requirements where we have a host of them. However, these conceptual models are sometimes hard to read. Because when a model has too many concepts, humans' working memory capacity limits our ability to comprehend, resulting in mental overload due to the substantially large set of stories modelled.

Data visualization is a great tool for making sense of complex data. Knowledge graphs like Neo4j possess tools for visualizing and exploring graph data that can help us find insights. This significantly helps impact stakeholders understanding in a greater depth, helping to showcase the relationships between the entities especially capturing user rationales, contained in a large set of stories in a visually descriptive and interactive manner. They help overcome existing approaches that fail to decompose elements within user stories in a highly illustrative way. Visualizations help facilitate the study of inter-dependencies of the entities contained therein, helping stakeholders hold more robust discussions on user requirements, aiding future change management processes, and facilitating rapid response to user needs. In addition, system integration processes essential to meet the rapidly changing business landscape requirements can be easily accomplished and explained to many audiences irrespective of their complexity.

The use of knowledge graphs helps shrink the semantic dissonance that can occur between how stakeholders conceptualization of a list of user requirements. Graph databases have an index-free adjacency performance feature, which make traversing a graph can be swift and independent of the data's overall size. As a result, queries can handle huge data sets easily without any constrains.

1.1 Structure

Section 2 examines the literature. The research concept and methodology are presented in Section 3. The approach and technical specifics of artefact development are described in Section 4 describes the approach and the technical details of artefact development. Section 6 assesses the artefacts using the research questions (RQs) in Section **??** and explains the RQs in Section 5. Section **??** examines risks to the study's authenticity and Section 7 brings the report to a close.

2 Literature Review

This section outlines the review of related research work concerned with Requirements Engineering (RE), Natural Language Processing (NLP), User Stories (US) and the visualization of its constituents elements. We organize our discussion under four headings, covering Requirement Engineering, Users stories and NLP Conceptual Models (Section 2.1), approaches for visualizing user stories (Section 2.2), and Data representation using knowledge graphs as a Visualisation tool (Section 2.3) where we detailed the gap our work would fill.

2.1 General Definitions

2.1.1 Requirement Engineering

In 2002, Marciniak [11] in his work defines requirements engineering as the systematic use of proven principles, techniques, languages, and tools for the cost-effective analysis, documentation, and ongoing evolution of user needs and the specification of the system's external behaviour to satisfy those user needs. In 2005, Pressman [12] stated that although requirements engineering may appear to be a straightforward process, that may be far from reality, as this notion may be misleading. Gotel et al. [13] in 2008 described it as the early activities in which stakeholders are identified, problems explored, and goals defined. It identified the process as the most data-intensive and media-rich aspect of software engineering and the period in which informal aspirations converge to an agreed statement of the problem and requirements specification. According to practitioners like Rasmusson [14], the documentation produced during the requirements engineering process captures the big picture, which acts as the framework for making decisions about the software development in question. While it is possible to express a requirement in many different ways as opined by Fraser et al. and Goel in their works in [15] and [16], respectively, our work focuses on requirements documented in the user story form obtained from various online sources. We rephrased some of these requirements as it is better to have more stories than to have stories that are too large as opined by Cohn in [17].

The requirement engineering process is heavily dependent on effective communication of user needs and system requirements between different stakeholders (such as requirements engineers, business analysts and users). As such, chances of misinterpretation of requirements or misinformation may arise, leading to a significant amount of time wasted on reworks of this process. We aim to proffer answers to whether knowledge graph's use in requirements engineering can help improve this process. As stakeholders are required reach a consensus when drafting these requirements that should capture user needs in ensuring the final development output meets these needs and the overall goal of the establishment.

2.1.2 Users stories and NLP Conceptual Models

As mentioned in section 1, the pursuit of developing solutions that can continuously adapt to the ever-changing business user's needs brought about the rise in the use of Agile Software Development approach as opined by Beck et al. in [5] in 2001. Various methods of the ASD adopts the use of user stories in capturing these requirements. As Cohn [17] puts it in his book titled: Stories Applied, For Agile Software Development, in addressing the challenge of constant change in user requirements and the abstractness of the entire software development process. Decisions about requirements are made based on the current information in an iterative manner using user stories. The writer defined a user story as a written description of the system functionality valuable to a user or purchaser of a system or software, traditionally hand-written on paper note cards.

In this section, we build on the discussion from the previous section 2.1.1. Having established the need for the use of user stories in RE. In here, we review prior literature works in the application of NLP for User Story analysis providing a basis for the elements of the stories we intend to visualise and how it impacts our research questions.

In 2012, Casamayor et al. in their state of the art review article in [18], affirmed that intelligent text analysis analysis tools like NLP provides computational support for requirement analysts to automatically or semi-automatically process all the information gathered when engaging users. Enabling them to classify, prioritize, determine the quality, translate to more formal specifications, and meet other requirement analysis tasks, suggesting that visualization of user story elements, relationship and entities content would further aid the stakeholders understanding of underlying conceptual models. As alluded by Cooper et al. in [19] earlier in 2009, that the use of visualization is beneficial in overcoming the challenges presented by traditional natural language requirements, suggesting that it can help convey more complex concepts such as the "health" of a set of requirements.

In 2020, to overcome the weakness of existing time-consuming manual approaches in building goal models from a list of User Stories. Gunecs et al. in [20] used advancements in Natural language processing to automatically create a goal model from a series of user stories, providing direct relations among goals captured in their relationships. While this is important, we aim to support analysts to efficiently address changes in user stories models by representing the stories in graph form that is queryable at any instance, helping to show relationships amongst entities contained in the list of user requirements.

2.2 Approaches for visualizing User Requirements

In section 2.1 we reviewed a couple of prior work focusing on RE, User Stories and how the the conceptual models created using NLP algorithms are of importance to practitioners. In this section, we examine pieces of prior literature that has adopted a model in visualising extracted constituents of user stories and their importance to stakeholders understanding. Our work focus on visualising the inter dependencies of User Stories on the basis of entities and relationship between such entities to aid granularity analysis through querying of the knowledge graph used in for its storage.

In 2016, Wautelet et al. [21] built an integrated ComputerAided Software Engineering (CASE) tool for the convenient graphical representation of User Story elements to aid the study of their refinements, compositions and decomposition in order to group them consistently. To create a similar technique to User Story Mapping used in previous research that would enable the ability of practitioners organise their User Stories on the basis of a proper granularity analysis.

In 2017, to address the drawbacks of employing people to identify entities and relationships in user stories and encourage conceptual models for requirement discussion. Lucassen et al. in [22] adopted the use of a Visual Narrator to extract conceptual models from user stories written in natural language, identifying dependencies, redundancies, and conflicts in a large set of requirements with a view of reducing the substantial human intervention required to obtain higher accuracy. The limitation of this tool was that it possessed a limited scope as it could not support a huge list of user stories dataset addressed by our proposed tool.

Researchers cannot overemphasize the importance of visualizations during software development and maintenance. It suggests areas of exploration when developers may not be aware of issues significant for user story understanding as opined by Paredes et. al. in [23].

2.3 Data representation using knowledge graphs as a Visualisation tool

To the best of our knowledge, prior literature work is absent on the use of knowledge graphs as a means of storing, querying, and visualising user story entities (who, what, relationships between entities and why) to aid better stakeholder's understanding and conversation towards meeting up in delivering solutions for overcoming the challenges of rapid change in business needs.

In 2017, Lu et al. [24] alluded that a type of knowledge graph called Neo4j was suitable for handling a large number of complex, dynamic, interactive, and low structured data. The writer applied the tool for Film data analysis to find the relationship between large data-sets of exponentially growing data. This result is significant because it helps resolve the large model information visualization problem as the user stories data-set grows. Dalpiaz et al., in their work in [25] published in 2019, identified this limitation in their prior work titled automated extraction of conceptual models from user story requirements: the Visual Narrator [22]. They resolved to use a Venn diagram to improve the situation in their later publication, which they believed was suitable for displaying overlapping user story elements. However, their work in [25] also creates tunnel vision that pushes people to focus on the colours and words without considering their context. In 2020, Cheng et al. [26] applied Neo4j to the disintegrated knowledge problem and time-consuming method of investigating Thangka graph icon characters: a kind of Tibetan painting art. The writer revealed that knowledge graphs were beneficial to the knowledge visualization and information retrieval of Thangka. The significance of Cheng et al's work is their ability to present entities and the semantic relationship between entities to aid the knowledge advancement of Tibetan painting art. Similar to the work, we aim to adopt the Neo4j to visualize entities and interdependencies contained within a list of user stories visualize entities to stakeholder's understanding.

3 Research Design & Methodology

In this section, we describe the "building blocks" that was used within this research. Our main objective is to develop an arbitrarily sophisticated model using a graph database that closely matches the narrative in our User Stories through connecting simple abstractions of nodes and relationships into connected structures that are simpler and more expressive than those earlier created in prior research work. A couple of concepts have been proposed in prior literature relating to agile and NLP for requirements engineering upon which we have built up our proposed model.

Here, we initially used an NLP package called Spacy to extract the different aspects (i.e., the WHO, WHAT and WHY) of our unstructured user story list and a graph data

storage technique to store and visualise them. However, this had some limitations; because similar entities in the "WHAT" and "WHY" aspects of the stories were classified under various labels. As such, To further identify the entities in these aspects, we used the APOC Plugin of the NEO4J, which uses Google's Cloud NLP package in enriching the entities identified from these aspects. The functional requirements for user stories used in our work for this technique are expressed as discussed in subsections 3.4 and 3.5. The illustrative framework in Figure 1 is depicts the processes followed in our work.

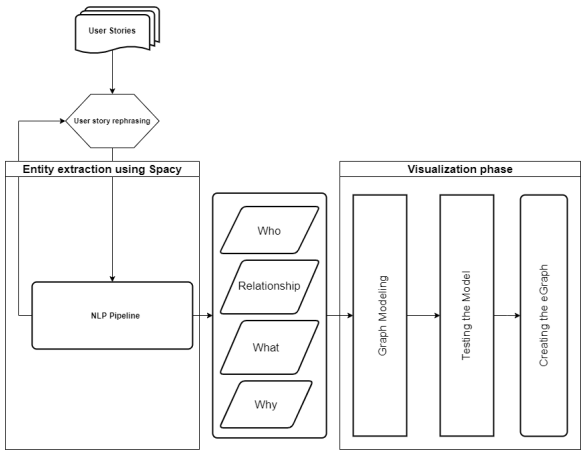

Fig. 1. Visualisation Framework

The section is divided into five subsections where we addressed: Research Questions in 3.1, Technique Selection in 3.2 The type and source of data used in section 3.3, Format used in rephrasing the each User Stories in section 3.4, and the NLP algorithm in section 3.5.

3.1 Research Questions

RQ1. How effective are knowledge graphs in visualizing user story conceptual models? To study the importance of using knowledge graphs in storing and visualising the user story models, we need to determine how the use of the tool can help visualise the critical components in user stories, the who, relationships, the goals and their rationals. RQ1 (answered in Section 6) aims to quantitatively analyze the effectiveness of knowledge graphs in user stories representation. Effectiveness is a measure of whether knowledge graphs use in user story conceptual models are suitable for the task(s), representation, and mapping based on the underlying data as highlighted by Knight, C., 2001 in [27].

RQ2. How is an effective visualization related to understanding the intrinsic features of a list of user stories? The extracted data from a list of user stories and the visualisation task that our project focuses upon impact stakeholders' understanding of how the entities and relationships within data are linked. Graph databases enable us to build arbitrarily sophisticated models closely mapped to our problem domain. As we argue in our answer to RQ1, they are generally effectively able to process complex data such as user stories that aids analysis and enable stakeholders to handle complex requirements effectively. We nevertheless see a degree of variation in performance in terms of storage and processing across our case studies.

3.2 Technique Selection

Our technique selection was experimental but subject to the storage and processing capacity of the knowledge graph tools. First, we were interested in using N.L.P. algorithm in extracting elements of User stories that are less complex or could be further broken down if they were epics into sub-stories. This criterion ensured that the stories the tools would analyse were as complete and straightforward as possible. Second, at this point, due to time limitations, we did not use the involvement from domain experts for building the domain models or performing any tasks. Our study is based on our work experience in the I.T. and Banking operations field.

3.3 Data Collection Procedure

We adopted a simple data collection procedure. We found a collection of 22 datasets of 50+ requirements each during our online search, expressed as user stories at [28]. We limited our research to just one of the datasets. The contributor on the page stated the datasets were all found online or retrieved from software companies with their permission to disclose. We rephrased some of these stories, and further broke down some others that we observed were too complex, as detailed in the section 3.4.

3.4 User Story rephrasing technique

As input, requirements in User Story form were rephrased using natural language notation without losing their contextual meaning. The most common template was adopted as described by [] is: "As a ***role***, I want ***goal***, so that ***benefit***".

For example, "As a ***Visitor***, I want ***choose an event***, so that I can ***buy tickets for that event***".

Where the:

1) ***Visitor*** is the ***Role***.
2) ***Choose*** is the relationship with the ***Goal***.
3) The ***Goal*** is to ***choose an event*** usually a containing a entity/object ***an event*** upon which an operation would be performed.
4) The ***benefit*** is to ***buy tickets for that event***.

We used at above template as base and rephrased the stories in that perspective following the pattern described below:

As a/an/the ***type of user/Role*** , I want *[*to*]* ***perform some operation connected components to the operation*** , *[*so that *[*I can*]* ***some reason*** *]*.

Where the:

1) ***[type of user/Role]*** is the ***Who*** usually a ***Noun*** or ***Noun Phrase***.
2) ***[perform some operation]*** is the ***Relationship*** usually a ***verb phrase*** .
3) ***[connected components to the operation]*** is the ***What*** usually a containing ***additional components***.
4) The ***[some reason]*** is the ***Rationale/Benefit***. The reason the requirement is needed. However, a story may not contain this aspect, as a user may consider it too mundane or generic to be included in their stories.

3.5 NLP algorithm pipeline

Using the technique in section 3.4, 70 user stories of the chosen dataset were rephrased. We adopted the use of Spacy N.L.P. package to extract the needed components (applying the tokenization, lemmatization and POS-tagging feature of the package on each story at various stages of our procedure). The input was a list of user stories stored in an excel file, loaded using the pandas dataframe. The code for the extraction processed could be examined in our github repository [1].

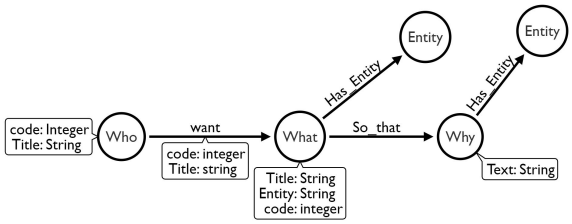

Fig. 2. Graph Model

4 Artefact Development Approach

In this section, we describe how our minimum viable artefact i.e. the visualisation was developed. The next section 4.1 is divided into three sub sections where we described the Graph Modelling in sub section 4.1.1, how the model was tested in 4.1.2 and Graph Creation in 4.1.3.

4.1 The visualisation

4.1.1 Graph Modelling

Conventionally, relational databases are used for any data warehousing and retrieval solution. However, their rigid schema and complex modelling features are not suitable for supporting rapid changes that can occur with storing unstructured text like our User Stories. The models created using relational databases result in an underlying storage model entirely different from the stakeholders' conceptual view of the application. The graph database is good at implementing conceptual models closely aligned with the application domain and not sacrificing performance or resulting in increased complexity as the data grows. It can maintain the integrity of the data as it undergoes rapid change due to changes in business needs and requirements growth.

Usually to begin modelling, an inexpensive approach such as whiteboard sketches of the entities contained in the list of the stories is adopted by requirements engineers, describing and agreeing upon the domain with the users. For example, we captured its appropriate who, what and why as labels with attributes such as properties, and connections to neighbouring nodes as relationships for each entity.

Also, we used the Awesome Procedures On Cypher plugin of the Neo4j - code-named APOC. This plugin allows us to call external NLP services for entity recognition, provided by cloud providers companies, like google and amazon, e.t.c, enabling us to identify the entities in the "What" and "Why" aspects of our decomposed user stories and how they are related. In our scenario, we would be working with the Google Cloud Platform Natural Language Processing API, see [2] for its documentation. This required us to have an API key used in connecting to the service.

In a graph database, what you sketch on the whiteboard is what you save in the database. We ensure that each node has suitable component specific labels and properties to fulfil its devoted data-centric obligations in graph terms, while, also making sure that every node is in the proper semantic context.

Nodes, relationship, and characteristics makes up the knowledge graphs core of our data model. Nodes are similar to object instances in the User Stories like the WHO, WHAT and WHY, as well as, the entities contained in them. Different nodes have labels and are connected by various relationships. For instance a Data User in Figure 3 below is a node while the Entity: Who is its label. The semantic context of each entity is further established by how these nodes are connected via the relationships that can have a type and direction. Each relationships has an ID attribute, while the WHAT node has an ID, type (signifying the details of the requirement) and the rationale or WHY where it exists see figure 4. A key-value pair is used to define the attributes of nodes and relationships.

Below in Figure 3 is how the following User Story without a rationale are modelled in the database. While examples of stories listed below with rationales, are modeled as shown in figure 4

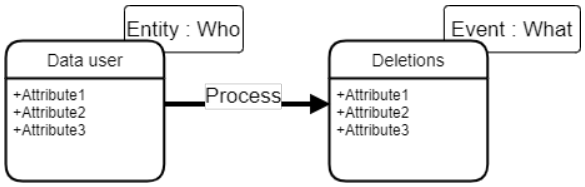

Fig. 3. As a Data user, I want to process deletions for 12-19-2017.

1) As a UI designer, I want to redesign the Resources page, so that it matches the new Broker design styles.
2) As a UI designer, I want to report to the Agencies about user testing, so that they are aware of their contributions to making Broker a better UX.
3) As a UI designer, I want to move on to the round 2 of DABS or FABS landing page edits, so that I can get approvals from leadership.
4) As a UI designer, I want to move on to the round 2 of Homepage edits, so that I can get approvals from leadership.

4.1.2 Model Testing

To mitigate poor design decisions, we reviewed the domain model and the resulting graph model at the early stage of our development, ensuring that subsequent changes to the graph's structure will be driven only by business

1. https://github.com/Ayodeji-python/SIT723/blob/main/\User_Stories_Extraction_Latest.ipynb
2. https://neo4j.com/labs/apoc/4.1/nlp/gcp/

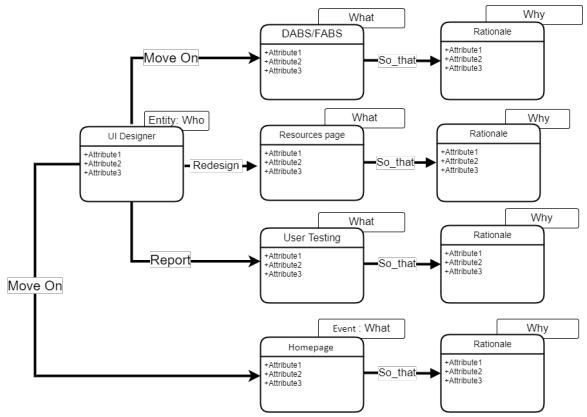

Fig. 4. Entity with more relationship nodes.

requirements changes rather than poor decisions during development.

To do this we adopted two approaches:

1) A simple one where we pick a start node and then follow relationships to other nodes, reading each node's labels and each relationship's name as we go, which we expect should create some rational understanding of the user story extracts. e.g., The sample in figure 4 would read as a U.I. designer redesigns the Resources page, reports User testing, moves on to the DABS/FABS page, and moves on to the home page.
2) Secondly, we adopted queryability approach, similar to the use cases we would apply on the graph when completed. We do this in ensuring the graph supports the kinds of queries we expect to run on it. For instance, one of our problem statements is that end-users report on how the entities and relationships within data are linked. To determine how entities are related, we need to be able to identify nodes that are connected to the node of interest and be able to visualise it. If we can craft a query that addresses this use case, we can be even more confident that the graph meets the needs of our domain. For instance, if we wanted to discover more about what a UI_Designer can do, we would start our query from the UI_Designer node. We do this by specifying the entity label WHO and the name property. The language that is specific to NEO4J, the knowledge graphs we adopted in this work is ***Cypher***. Cypher is an expressive graph database query language [29].

Listing 1. Query

```
MATCH (w:Who)-[:WANT]->(b)-[:So_that]->(c)
WHERE w.name = 'UI_designer'
RETURN w,b,c
```

Starting from a user who is a ***"UI designer"***, we ***MATCH*** against the ***WHO*** in the graph along a directed path of all its ***WHAT*** with the WANT relationship and then along all related ***WHY***. The ***RETURN*** ensures that the ***WHO***, ***WHAT***, and ***WHY*** are returned in the results. With such a query readily supported by our graph, we are assured that the design fits its purpose.

4.1.3 Graph Creation

We created the graph database for the extracted list of user stories dataset stored in our online repository[3] using the NEO4J desktop version. However, the online sandbox at see [4] can also be used. Each role i.e WHO in user stories are usually unique, we created a who_id feature for each role.

who_id	WHO	Relationship	Entity	WHAT	WHY
4	Data user	process	deletions	process deletions for 12-19-2017	
9	UI designer	redesign	Resources page	redesign the Resources page	it matches t
9	UI designer	report	Agencies	report to the Agencies about user testing	they are awa
9	UI designer	move	DABS or FABS landing page edits	move on to the round 2 of DABS or FABS landing page edits	get approva
9	UI designer	move	Homepage edits	move on to the round 2 of Homepage edits	get approva
9	UI designer	move	Help page edits	move on to the Help page edits on 3rd round	get approva
5	developer	log	events / activities	log events/activities better	troubleshoo
5	developer	update	FABS submission page	update the FABS submission page when the publishStatus changes	know when

Fig. 5. Data set

To enforce this, a unique constraint on the role was created using the code snippet below.

Listing 2. Unique Constraint on roles

```
CREATE CONSTRAINT ON (w:Who)
ASSERT w.code IS UNIQUE
```

To load the data set containing the extracted components from the repository, thereby creating the graph database. We used the code snippet below. The LOAD CSV WITH HEADERS command loads the data set stored from the repository where it stored, identifying

Listing 3. creating the WHO label on roles

```
LOAD CSV WITH HEADERS FROM
'https://raw.githubusercontent.com/
Ayodeji-python/SIT723/main/new_data.csv'
AS row
WITH DISTINCT row.who_id AS who_id,
row.WHO AS Who
MERGE (w:Who {code: toInteger(who_id)})
ON CREATE SET w.name = Who
```

Listing 4. Creating the WHAT node for each WHO

```
MATCH (w:Who {code: toInteger(row.who_id)})
CREATE (wh:What)
SET wh.name = row.Entity,
    wh.Text = row.WHAT
CREATE (w)-[:WANT {title:row.Relationship}]
->(wh)
```

Listing 5. Creating the WHY node and relationship with WHAT

```
CREATE (r:Why)
SET r.name = row.Entity,
    r.Text = row.WHY
CREATE (wh)-[:So_that]->(r)
```

3. https://raw.githubusercontent.com/Ayodeji-python/SIT723/main/new_data.csv
4. https://sandbox.neo4j.com/

5 Results & Discussion

This section showcases the results of our work in 5.1 and carried out some analysis which the tool can be used for in 5.2.

5.1 Results

Although not all the contents of the nodes and arc are visible, we have tried to display the whole visualization of the Graph database of the entire list of user stories used in this research via [5]. This is to enable our audience have a grasp of what is being described and how the structure looks like, which is further analysed in 5.2.

The experiment has a sample of 70 user stories which consisted of eleven distinct roles, i.e. WHO represented in red circles. The brown and ash circles represent the WHAT and WHY respectively. While, others are entities extracted using the NEO4J APOC Plugin. These entities are obtained from the WHAT and WHY aspects of the stories. The lines between the circles are the relationships between the components. Although, the visualization may look clumsy and less descriptive. However, our ability to query it to visualize the different aspects to aid analyst's work in obtaining more insights into the connections between the entities is the focus of our study. In the next section, we will be discussing the fundamental way of analyzing our user story data.

5.2 User Story Analysis

Following the creation of a database of user story data, we will begin the user story analysis by focusing on the following aspects:

1) The attributes of a WHO label(s) will be examined;
2) The relationship between two or more nodes; and,
3) Using keywords extract hidden insights.

The fundamental way for analyzing our user story data is to explore and query the graph database. The Neo4j graph database focuses on improving performance in standard RDBMS queries adopting a large number of links between nodes. The database traverses the nodes and edges at the same rate, and the ergodic speed is not degraded with increase in graph data volume [30].

There are several clauses in Cypher. However, ***MATCH*** and ***WHERE*** are two of the most prevalent.These routines differ slightly from SQL functions. ***MATCH*** is used to describe the pattern's structure, which is mostly based on relationships. ***WHERE*** is a pattern constraint that is used to add more constraints to patterns. There are other clauses for writing, modifying, and deleting data in Cypher. To construct and delete nodes and relationships, use the ***CREATE*** and ***DELETE*** commands [24].

5.2.1 *Querying nodes*

The query of nodes attributes is required when we just know more such node. For instance in this example, we wish to know a user's role, i.e. WHO and its features .Using the Cypher code below in listing 6 we can get the information about a single WHO node. This query returns the details about the WHO for the role of a "UI Designer" as shown in figure 6.

Listing 6. Query for single WHO

```
MATCH (n:Who)
WHERE n.name = "UI designer"
RETURN n
```

Fig. 6. Details of a WHO label

Also, more connections to a node can be revealed when the expand/collapse toggle button is clicked upon, when the a query is run. See 7.

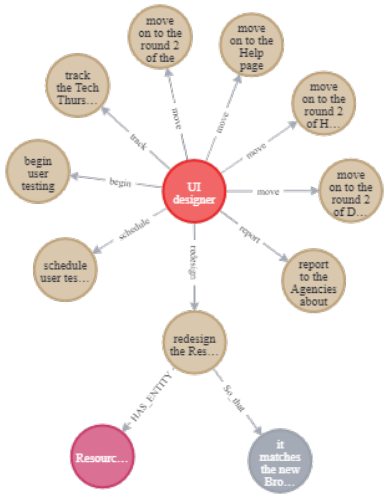

Fig. 7. More node connections

5.2.2 *Querying relationship between two or more nodes*

By understanding who wants what. How users are connected to what they want, and who is responsible for what. We generate tremendous insight into the unseen forces that influence the effective use of a system. A graph database is good at dealing with a large number of Complex, interconnected data. When we want to know the node that has a relationship with a known node, we can query the relationship between the two nodes. For example, where we want to find the list of what a user who is a "Developer" wants. The query in listing **??** is very useful because an analyst or a requirements engineer may want to know in a holistic details what a user requires of a particular user or a set of users. See visualisation in figure 8

Listing 7. Query for relationship MATCH (n:Who)-[:WANT

```
-[:So_that|HAS_ENTITY]->(m:Why)
-[:HAS_ENTITY]->(k:Entity)
WHERE n.name = "developer"
RETURN n,wh,m,k
```

5. https://github.com/Ayodeji-python/SIT723/blob/main/whole_graph.png

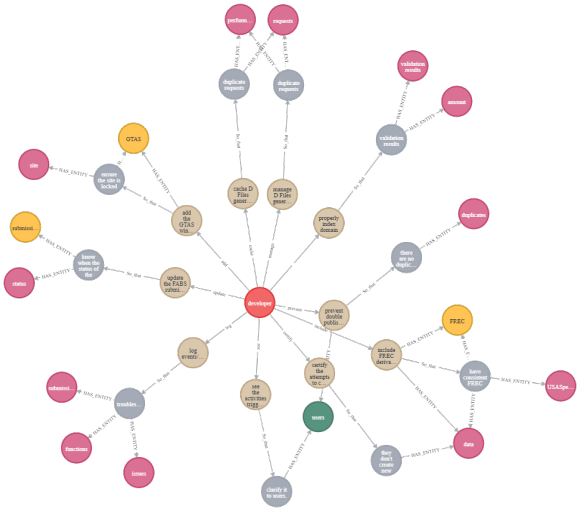

Fig. 8. Relationship features between WHO, WHAT and WHY

5.2.3 *Using keywords*

There are scenarios where analysts only remembers a word or phrase about something to search the graph during an investigation. For instance, consider a scenario where an analyst seeks to know all users who have something to do with an application, say FABS (one of the applications identified in our user stories). The analyst can use the cypher code in the listing 8 to extract such information. This would return the visualisation in figure 9.

Listing 8. Query using keywords

```
MATCH (n:Who)-[:WANT]->(wh:What)-[:HAS_ENTITY]
->(m:Entity)
WHERE wh.name CONTAINS 'FABS'
RETURN n,wh,m
```

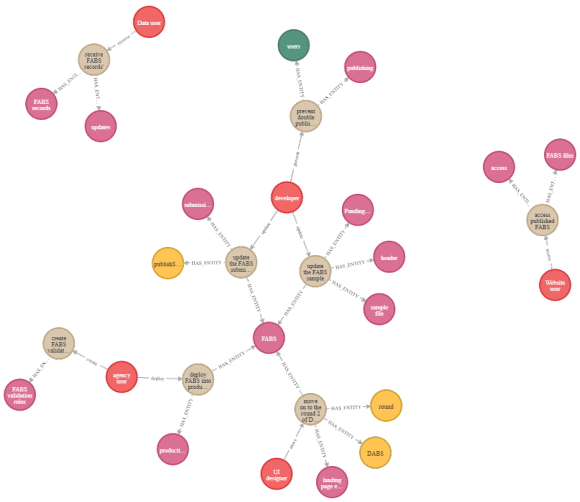

Fig. 9. Using Keywords

Other clauses such as ***STARTS WITH***, ***ENDS WITH***, e.t.c can be used to match string keyword in the WHERE statement of the Cypher code as in listing 8 to help analysts search through the graph database using keywords or regular expressions. Thus, facilitating easy filtering and limiting the visualization returned to what humans can easily comprehend. To see more of the clauses visit https://neo4j.com/docs/cypher-manual/current/clauses/where

6 Empirical Evaluation

We plan on using human subjects who are business analyst to evaluate the outcome of the research. However, due to the limitation of time and the negative impact of COVID-19 this was not possible.

7 Conclusion & Future Work

A graph database, such as NEO4J, is ideal for database development and dynamic information visualization. It allows querying with or without limitations, allowing an analyst to grasp the user story components immediately. In this work, we have been able to use the tool on user stories whose components were extracted using NLP Algorithm to help analysts further understand the roles, wants and rationales in a visually descriptive manner.

7.1 Future Work

The limitation of this current study is the some entities contained in the WHAT and WHY aspects of the users stories are still left out as revealed in 9. This limitation could be a source of future research work in revealing insights from extracted users stories components. However, considering the data model we have used here, the output helps analysts understand the roles played by each user with a view of ensuring business needs are met in an efficient and effective manner.

Acknowledgments

I would like to thank my Supervisors, Dr. Chetan Arora and Dr. Hourieh Khalajzadeh of Monash University for providing me with the needed guidance during this work. I also wish to acknowledge the support of my wife, kids and my colleagues at the Central of Nigeria.

References

[1] I. ISO, IEEE, Systems and Software Engineering–vocabulary, IEEE Computer Society, Piscataway, NJ, 8 (2010), p. 9.
[2] P. Bourque and R. E. Fairley, eds.,SWEBOK: Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, Los Alamitos, CA, version 3.0 ed., 2014.
[3] D. Spath, F. Hermann, M. Peissner, and S. Sproll, User requirements collection and analysis, Handbook of human factors and ergonomics, (2012), pp. 1313–1322.
[4] G. Lucassen, Understanding User Stories: Computational Linguistics in Agile Requirements Engineering, PhD thesis, Utrecht University, 2017.
[5] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al., Manifesto for agile software development, Information and software technology, (2001).
[6] E.-M. Schön, J. Thomaschewski, and M. J. Escalona, Agile requirements engineering: A systematic literature review, Computer Standards & Interfaces, 49 (2017), pp. 79–91.
[7] T. Dybå and T. Dingsøyr, Empirical studies of agile software development: A systematic review, Information and software technology, 50 (2008), pp. 833–859.
[8] V. One, 11th annual state of agile survey, Acessado em, 6 (2017).

[9] E.-M. Schön, J. Thomaschewski, and M. J. Escalona, Agile requirements engineering: A systematic literature review, Computer Standards & Interfaces, 49 (2017), pp. 79–91.
[10] D. ISO, 9241–210: 2010: ergonomics of human-system interaction—part 210: human centred design for interactive systems (formerly known as 13407), Switzerland: International Standards Organization, (2010).
[11] J. J. Marciniak, Encyclopedia of software engineering, John Wiley & Sons, Inc., 2002.
[12] R. S. Pressman, Software engineering: a practitioner's approach, Palgrave macmillan, 2005.
[13] O. C. Gotel, F. T. Marchese, and S. J. Morris, The potential for synergy between information visualization and software engineering visualization, in 2008 12th International Conference Information Visualisation, IEEE, 2008, pp. 547–552.
[14] J. Rasmusson, The agile samurai: How agile masters deliver great software, Pragmatic Bookshelf, 2010.
[15] M. D. Fraser, K. Kumar, and V. K. Vaishnavi, Strategies for incorporating formal specifications in software development, Communications of the ACM, 37 (1994), pp. 74–87.
[16] V. Goel, Sketches of thought: A study of the role of sketching in design problem-solving and its implications for the computational theory of mind, PhD thesis, University of California, Berkeley, 1991.
[17] M. Cohn, User stories applied: For agile software development, Addison-Wesley Professional, 2004.
[18] A. Casamayor, D. Godoy, and M. Campo, Mining textual requirements to assist architectural software design: a state of the art review, Artificial Intelligence Review, 38 (2012), pp. 173–191.
[19] J. R. Cooper Jr, S.-W. Lee, R. A. Gandhi, and O. Gotel, Requirements engineering visualization: a survey on the state-of-the-art, in 2009 Fourth International Workshop on Requirements Engineering Visualization, IEEE, 2009, pp. 46–55.
[20] T. Güneş and F. B. Aydemir, Automated goal model extraction from user stories using nlp, in 2020 IEEE 28th International Requirements Engineering Conference (RE), IEEE, 2020, pp. 382–387.
[21] Y. Wautelet, S. Heng, M. Kolp, I. Mirbel, and S. Poelmans, Building a rationale diagram for evaluating user story sets, in 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), IEEE, 2016, pp. 1–12.
[22] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. Van Der Werf, and S. Brinkkemper, Extracting conceptual models from user stories with visual narrator, Requirements Engineering, 22 (2017), pp. 339–358.
[23] J. Paredes, C. Anslow, and F. Maurer, Information visualization for agile software development, in 2014 Second IEEE Working Conference on Software Visualization, IEEE, 2014, pp. 157–166.
[24] H. Lu, Z. Hong, and M. Shi, Analysis of film data based on neo4j, in 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), IEEE, 2017, pp. 675–677.
[25] F. Dalpiaz, I. Van Der Schalk, S. Brinkkemper, F. B. Aydemir, and G. Lucassen, Detecting terminological ambiguity in user stories: Tool and experimentation, Information and Software Technology, 110 (2019), pp. 3–16.
[26] S. Cheng, T. Wang, X. Guo, and Y. Wang, Knowledge graph construction of thangka icon characters based on neo4j, in 2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), IEEE, 2020, pp. 218–221.
[27] C. Knight, Visualisation effectiveness, in 2001 International Conference on Imaging Science, Systems, and Technology (CISST 2001), Citeseer, 2001, pp. 249–254.
[28] F. Dalpiaz, Requirements data sets (user stories), 2018.
[29] I. Robinson, J. Webber, and E. Eifrem, Graph databases: new opportunities for connected data, " O'Reilly Media, Inc.", 2015.
[30] W. Yulan, Comparison of graphic database neo4j and relational database, Modern Electronics Technique, 20 (2012), pp. 0077–03.

Ayodeji Ladeinde is a current Masters degree student at Deakin University, Melbourne, VIC, Australia. Before embarking on his degree, he worked as a secondary market operations specialist at the Central Bank of Nigeria for about a decade. He was involved in the administration of the Bank's Scrip-less Securities Settlement System's and also managed the Real-Time Gross Settlement System, enabling daily settlement of cash and securities transactions of over 25 deposit money banks with over 75 million customer base. He delivered bespoke secondary market products and services to Nigerian banks and implemented Monetary policy directives passed by the Bank's Monetary Policy Committee from time to time. He had his Bachelors degree in Mathematical Sciences (Computer Option) and a Post Graduate Diploma from the University of Agriculture in Ogun State, Nigeria and University of Lagos, Lagos, Nigeria, respectively. https://www.linkedin.com/in/ayodeji-ladeinde-b8469b122/