GENERATING GOAL MODELS FROM USER STORIES

by

Tuğçe Güneş

B.S., Computer Engineering, İstanbul Bilgi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

I want to express my gratitude to all people that helped and supported me throughout the process of writing this thesis. First of all, Fatma Başak Aydemir, who always thinks along and consistently provided me with feedback on the methods used. Second, to Tülay Güneş, who supported me my whole life and was an amazing role model as a mom. Third, my family, for sharing their love and support with me. Lastly, Tunga Güngör and Savaş Yıldırım for being my jury and participating in the defence committee.

ABSTRACT

GENERATING GOAL MODELS FROM USER STORIES

Natural language (NL) is used to express stakeholder requirements since it requires less time and energy. There are several ways to collect requirements, and user stories are an example of a semi-structured format. They are commonly used to capture user needs in agile methods due to their ease of learning and understanding. However, user stories can be large enough which makes it difficult to read and understand the relations among them. Such relations make it easier for developers to understand the structure of the project. Goal models, on the other hand, provide high-level perspective and explicit relations among goals but they require time and effort to build. First, we conduct an experiment to show the usefulness of the goal models for reading and comprehending the data set. This thesis proposes a goal model builder tool to automatically generate a goal model from a set of user stories by applying natural language processing (NLP) techniques. We first parse and store the extracted information from a set of user stories in a graph database to maintain the relations among the roles, actions, and benefits mentioned in the set of user stories. We create the goal model strategies using the information in the graph database, which enables us to see the connections between the nodes and edges. By applying NLP techniques and several heuristics, we produce goal models that resemble human-built models. Second, we contribute an evaluation of the goal model builder tool that determines whether the ArTu tool speeds up the creation of goal models. A cross-over experiment has been carried out to evaluate the time difference between a goal model with the tool and one without the tool. We put out a variety of hypotheses to contrast experiment findings. The results of several statistical analyses run on the experimental data show that the ArTu tool significantly reduced the time needed to build goal models.

ÖZET

KULLANICI HİKAYELERİNDEN AMAÇ MODELLERİ OLUŞTURMA

Doğal Dil (NL), daha az zaman ve enerji gerektirdiğinden paydaş gereksinimlerini ifade etmede kullanılır. Gereksinimleri toplamanın birkaç yolu vardır ve kullanıcı öyküleri yarı yapılandırılmış bir format örneğidir. Kullanıcı ihtiyaçlarını anlamak için kullanılırlar ve öğrenme-anlama kolaylıkları nedeniyle çevik yöntemlerde kullanılırlar. Ancak, kullanıcı öyküleri yeterince büyük olabilir ve bunlar arasındaki ilişkileri okumak ve anlamak zorlaştırabilir. Bu tür ilişkiler, geliştiricilerin projenin yapısını anlamasını kolaylaştırır. Öte yandan, hedef modeller yüksek düzeyli bir bakış açısı ve hedefler arasında açık ilişkileri sağlar ancak oluşturmak için zaman ve çaba gerektirirler. Bu tez, veri kümesini anlamak için hedef modellerinin yararlılığını göstermek için bir deney yapmakta ve doğal dil işleme (NLP) tekniklerini uygulayarak bir kullanıcı öyküsü setinden hedef modeli otomatik olarak oluşturmak için bir hedef modeli oluşturma aracı önermektedir. İlk olarak, bir kullanıcı öyküsü setinden çıkarılan bilgileri çözümlüyor ve saklıyoruz ve roller, eylemler ve faydalar arasındaki ilişkileri kullanıcı öyküsü setinde belirtilen bir grafik veritabanında koruyoruz. Grafik veritabanındaki bilgileri kullanarak hedef model stratejileri oluşturuyoruz, bu da düğümler ve kenarlar arasındaki bağlantıları görmemizi sağlıyor. NLP tekniklerini ve birkaç heuristiği uygulayarak, insan yapımı modellere benzeyen hedef modelleri üretiyoruz. İkinci olarak, ArTu aracının hedef modeli oluşturma hızını artırıp artırmadığını belirleyen bir değerlendirme sunuyoruz. Aracı kullanarak ve kullanmadan önce hedef model arasındaki zaman farkını değerlendirmek için bir çapraz deney yapıldı. Deney bulgularını karşılaştırmak için çeşitli hipotezler ortaya koyduk. Deneysel veriler üzerinde çalıştırılan birkaç istatistiksel analizin sonuçları, ArTu aracının hedef modellerini oluşturmak için gereken zamanı önemli ölçüde azalttığını göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

AGORA	Attributed Goal-Oriented Requirements Analysis Method
GBRAM	Goal Based Requirements Analysis Method
GORE	Goal-Oriented Requirements Engineering
GRL	Goal-Oriented Requirement Language
GQM	Goal Question Metric
KAOS	Keep All Objectives Satisfied
MCES	Maximum Common Edge Subgraph
NFR	Nonfunctional Requirements
NL	Natural Language
NLP	Natural Language Processing
POS	Part of Speech Tagging
RE	Requirements Engineering
UML	Unified Modeling Language

1. INTRODUCTION

The most popular method for managing software requirements is to describe them in natural language (NL) [1]. Natural language documentation of stakeholder requirements takes less time and effort than alternative formal or semi-formal representations because it is more effective in eliciting requirements from stakeholders [2]. Requirements can be gathered in a variety of ways, including free-form, semi-structured, and structured formats.

User stories, a semi-structured natural language notation that is frequently employed in agile development methods [2, 3]. The easiest to learn and use a standard pattern for user stories is "`As a` *role*, `I want` *action*, `so that` *benefit*" [4]. User stories can be used in agile development because of these characteristics, which are also the major factors in practitioners' adoption of user stories.

User stories do not address the problem of capturing the structure of the user needs. A list of user stories in a flat format does not explain the purpose of the user stories or how they relate to one another. However, it is generally challenging to have a high-level picture of the relationships, identifying the user stories that are about the same concepts or declaring the same advantages even though filtering and sorting procedures can offer a rough overview.

Goal models can be used to represent the structure of a set of requirements and many distinct relations, as has already been shown in a large number of papers [5]. A goal modelling language can be used to specify numerous inter-dependencies, such as cost and value contributions [6], and high and low-level goals are associated through refinements. Actors' goals can also be visualized within the actor's boundaries [7].

Goal models are also used for analysis and optimization [8]. Despite extensive research on goal models, there is little industry adopted these models. The learning curve is considered to be high by practitioners, and it is often believed that creating goal models is time and labour-intensive and therefore expensive.

By helping them in taking advantage of the benefits that goal modelling offers, we hope to promote agile development processes that rely on user stories. To do so, we have built a goal model generator tool to automate the process of creating goal models using NLP features. Moreover, the tool is evaluated with a cross-over experiment, which proves that using the tool actually shortens the time to build a goal model, and provides more consistent design in terms of a number of model elements. Our previous work on extracting goal models from user stories has been published in two papers.

The contributions of this theses are as follows:

- We first experiment to see if the goal models improve the understanding of relations among user stories. The statistical tests have shown that when a user is asked a question about the goal model, the time to find an answer from a given goal model is significantly less than when a user story set is given.
- We build 5 heuristics to build goal models. There are multiple ways to draw a goal model based on people's points of view about the user story set. So these heuristics give an idea of 5 different strategies to generate a goal model. Our first 3 heuristics consider the actor and action parts of the user story, whereas the last 2 heuristics deal with combining the action part with the benefit part of the user story. By this, we try to achieve more complex designs, so that the goal model is more similar to human-built models.
- To make practitioners able to use our 5 heuristics and have automatically generated goal models, we have built ArTu tool. The tool is served to be reachable for our users, and it is designed to be user-friendly by helping users to upload a user story set in text format. After selecting the file to upload, they can choose a strategy and click the "generate" button. The tool also provides an interac-

tive page to make changes to the drawn goal model, download their data, and re-upload their last work.

- Evaluations on experiment results indicate that the editing time spent on goal models between tool-generated ones and manually generated ones are significantly different from each other.

The remainder of this thesis is structured as follows. Chapter 2 explains the background of user stories and goal models. Chapter 3 presents our motivation to create the ArTu tool. Our approach to design heuristics for generating goal models in Chapter 4 is published in [9], and the ArTu tool with its architecture as presented in Chapter 5 is published in [10]. Evaluation of the tool shown in Chapter 6, and threats to our work are discussed in Chapter 7. The feedback from participants is discussed in Chapter 8. Chapter 9 reports on related literature and Chapter 10 concludes our approach and ideas with future work.

2. BACKGROUND

We present the background of user stories and goal models in this chapter since in this thesis we integrate those concepts by creating a tool which consumes user stories in order to derive goal models at the end.

2.1. User Stories

The most common artifact in agile software development for expressing users' requirements is user stories [11]. A user story is a brief, straightforward description of a feature that is written from the viewpoint of the user or customer of the system who wants the new capacity.

User stories have become increasingly popular in the industry as a result of the widespread use of agile software development techniques like Scrum and Kanban [12]. They have been gaining a lot of attention as they help overcome the communication problem within a software development process by bridging the gap between those who want the software and who build the software.

In agile development techniques, user stories, a semi-structured natural language notation, are widely used. They are easily understood by stakeholders since they are written in natural language. Utilizing templates, user story generation is standardized. The most popular template for user stories was determined to be the Connextra template. The template takes the following form: "As a *role*, I want *action*, so that *benefit*". `Role` specifies a system user who wants the system to accomplish something, `goal` outlines the action system takes to help the user, and `benefit` explains why the action should be taken from the user's point of view [13].

This user story format significantly improves the why and how knowledge that developers must have to design a system that satisfies the needs of the users. It

describes the problem domain, the necessity for the activity (business value), and the user's "requirement" (activity), and offers a user-first perspective (role) [14]. For example, a student wants to upload homework through a system: *"As a student (*role*), I want to upload my homework (*what I want to do with the system*), so that I get good grades (*value I receive*)."*

User stories are used in different software development approaches like requirements elicitation, gamification, and agile development. In one of the gamification work [15], user stories are chosen as a method for requirements documentation due to their clarity, simplicity, and widespread use in agile development. Besides their simplicity, without any knowledge of notation or modelling, stakeholders can easily learn how to use them.

2.2. Goal Models

A goal is a desired state of affairs. In requirements engineering (RE) a goal is used to capture a stakeholder objective for the system to achieve. A goal model is a conceptual model where goals are interconnected with each other through relations, such as refinement. Goal models link high-level goals to low-level system requirements by displaying goals in a hierarchical sequence. Goals can be either functional or non-functional, functional goals outline the tasks the system should accomplish whereas non-functional ones show ideal system characteristics.

Various formats can be used to express goals e.g. casual, semi-formal, and formal approaches. Informal approaches typically utilize natural language text to communicate goals; semi-formal approaches mainly employ box and arrow diagrams, and formal approaches use logical assertions in some formal specification language [16].

Examples of goal modelling methodologies, or frameworks are KAOS, GBRAM, AGORA, NFR, i*, Tropos, and GRL [17]. KAOS, GBRAM and NFR frameworks try to specialize requirements by linking system components that are functional and

non-functional to business objectives, whereas i* do requirements elicitation to identify the need for change and the current state of the organization [16]. Figure 2.1 is an example of a goal model using the iStar framework. It has 2 sub-goals to reach the parent goal which is student actor wants to get good grades which require her to upload her homework and attend classes.

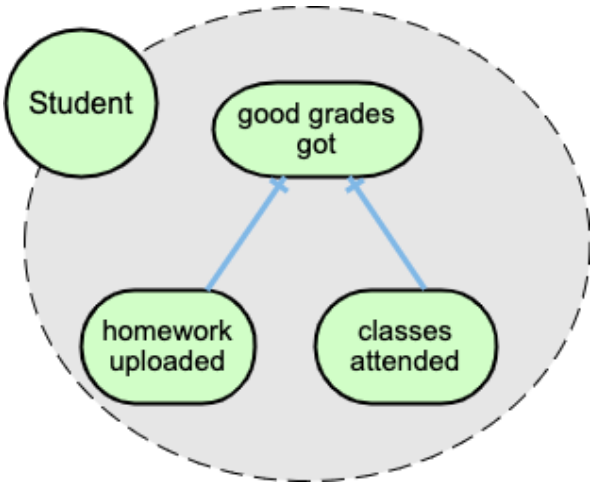

Figure 2.1. Example goal model.

Goal modelling is a common strategy in RE that helps capture stakeholders' needs and expectations as well as for determining whether and why software development is necessary. Goal models are deliberate depictions of user goals and potential strategies for achieving them. They are applied during the preliminary requirements analysis stage to clarify the reasoning behind a software system. They offer helpful structures to examine high-level goals and methods, and they have been used to describe the reasoning of both people and software systems [18].

Goal-oriented requirements engineering (GORE) is the study of or use of goal models in requirements engineering [19]. In RE research, GORE has drawn a lot of interest as a method for comprehending the underlying causes of system requirements, assisting in ensuring that the proper system is created to tackle the right issues [17]. There has been a lot of research on goal models, but little industry adoption. Practi-

tioners typically feel that there is a steep learning curve unlike user stories and that developing goal models requires a lot of time and labour, which makes them expensive.

3. MOTIVATION

Our motivation is explained in this chapter. To motivate the approach, we run an initial experiment on testing the use of goal models. We must establish the goals of measurement to choose relevant metrics for this experiment. The Goal-Question-Metric paradigm created by Basili et al. [20] can be used as a framework to construct metrics that support particular goals.

For this work, our *quality* is "time efficiency", and *metric* is "time difference in seconds" when goal models or user stories are used to answer certain questions about the user story set. Therefore, we build hypotheses to assess our time metric. Our null hypothesis is:

- H_time_0: The time to answer data set questions from user stories and to answer from goal models are equal.

The alternative hypothesis is:

- H_time_1: The time to answer data set questions from user stories and to answer from goal models are not equal.

3.1. Experimental Design

Purpose. We argue that while user stories are a great tool for connecting stakeholders and developers, it can be challenging to read a group of user stories in their raw form and understand how they relate to one another. However, goal modelling makes it possible to analyze user stories and ease comprehend how they relate to each other. In this initial experiment, we test the effectiveness of goal models to determine if they speed up the process of finding user story dependencies and improve readability.

Experiment Protocol. In this experiment, we want to perceive whether goal models help understand the software product needs. To assess the usefulness of goal models, we have a cross-over designed experiment with two groups a treatment group and a control group. There are two sessions in the experiment and two user story sets accordingly. In the first session, the control group is given a goal model drawn from one user story set and a set of questions for this data set, while the treatment group is given the other set of user stories in raw text format and questions related to it. The crossover experiment design then reverses the order of responding questions from a goal model or raw text during the second session. In other words, the second user story set and accompanying questions are given to the control group. While the first set of user stories and their questions are given to the treatment group. Table 3.1 details the cross-over design we used for this study.

Table 3.1. Experiment 1 setup design.

	User Story Set Given	**Goal Model Given**
User Story Set 1	Group 1	Group 2
User Story Set 2	Group 2	Group 1

In case they are unfamiliar with user stories and goal models, we offer them a brief introduction to these concepts at the beginning of the experiment. After briefing them on the experiment's methodology, we ask them to open the questionnaire form and begin with one set of user stories while sharing their screen. When people wish to look up words in user stories or goal models that are in text format, the search option is also available.

They are urged to answer questions aloud, and we time how long it takes them to do so. By doing this, we hope to determine whether or not they prefer the goal model or the text file for finding the answers.

After answering all questions, participants fill out their demographic information with their agreement after each session, which lasts about 20 minutes and requires a total of 40 minutes for one participant. The visual explanation of how each participant takes part in the experiment is shown in Figure 3.1.

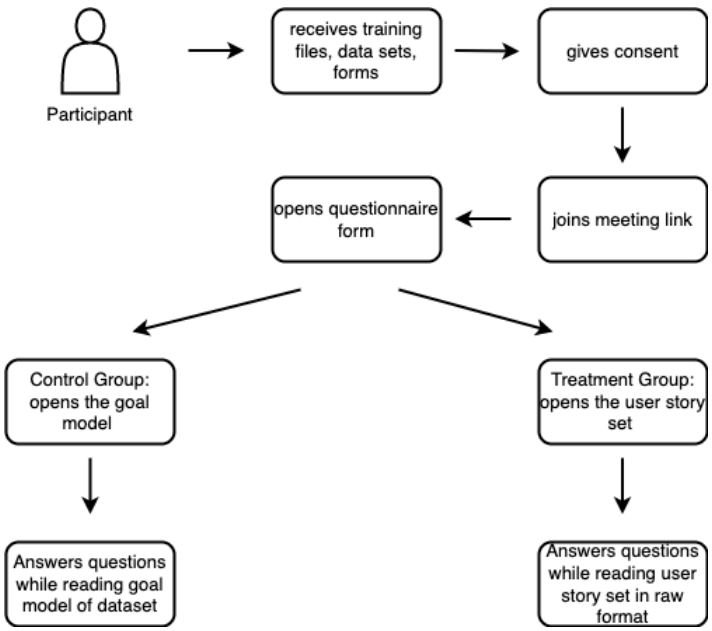

Figure 3.1. Experiment 1 flow.

Materials. Two user story sets are considered in this experiment, which is chosen in terms of being an understandable topic for anyone. We also need goal model designs for each user story set. So that our first step is to build goal models from these user stories. After having fully connected goal models using Istar_Stencil.vss [21], we created a questionnaire form for two user story sets in order to measure how well a participant understands the topic.

Table 3.2 displays a list of questions made regarding the Alfred set and table 3.3 for the Camperplus user story set during the experiment. Long-answer text format for the responses is anticipated. While examining the raw user story set or the goal model of this set, each participant responds to these two sets of questions.

Additionally, each participant receives instruction on how to create goal models and compose user stories, as well as information on the iStar language and the goal model's actor, actor boundary, goal, and refinements. The objective of this training is to ensure that all participants in the experiment have the same level of understanding. Only takes about 10 minutes and has 15 pages. The participant is encouraged to ask any questions they may have prior to the experiment after this teaching session.

Table 3.2. Alfred set questions.

Data Set Questions
What are the actions to provide 'usability' for the 'developer' role?
What can be done to supply data privacy according to 'Older Person'?
What are the vital signs to be monitored according to 'Medical Care-Giver'?
What are the actions to provide 'medicine intake' according to 'Older Person'?
What does 'developer' want to know when the application stops functioning?
What are the household items to control according to 'Older Person'?

Two user stories are used in the first experiment and they are both taken from [22]. One of the data sets, ALFRED, which focuses on creating a wearable device to track elderly people's data, contains 80 user stories now instead of the original 130, some of which were eliminated for being overly simplistic.

Another set, Camperplus, contains 55 user stories regarding a children's camping application. We manually draw the goal models for each set of user stories since, as was previously indicated, we also need the designs for these sets' goal models.

Table 3.3. Camperplus set questions.

Data Set Questions
What are the actions to achieve inner peace according to the 'parent'?
What are the camp administrator's actions to handle the behavioral issues?
Who is in charge of reporting to the manager if a camper has unseemly behavior?
What are the actions to inform families about their children according to 'camp administrator'?
What can be done to schedule the camp program according to 'camp administrator'?
What should be erased to get rid of redundancy according to 'camp administrator'?

There are 4 actors in the *ALFRED* dataset, and OlderPerson has more user stories than the other actors. Additionally, Medical Caregiver, Social Caregiver and developer actors exist in the set. We have 79 goals, 78 "and" refinements, 2 quality, and 7 help contributions in its goal model. For *Camperplus* dataset, there are 4 actors, with the Camp Administrator having more user stories than the other actors Parent, Camp Worker, and Camp Counselor. There are 69 goals and 43 "and" refinements in its goal model. In Figure 3.2, Parent actor with its goals and refinements are shown in the goal model, while in Figure 3.3 Medical Care Giver actor and its goals and refinements are presented.

What is Measured. What we want to measure in the first experiment is how much time each participant spends on each survey question to find the right answers. It is important to calculate the average time spent on every question for both when the user story set and the goal model is given respectively. As well as measuring the time, we also want to observe how accurately participants respond to questions. To be clear about the right answers, there is an example question regarding the Parent actor in Figure 3.2. *"What are the actions to achieve inner peace according to the parent?"* Camperplus set mentions inner peace within the parent actor in 5 goals. So to have an accurate answer, the participant is expected to write down all of the 5 goals.

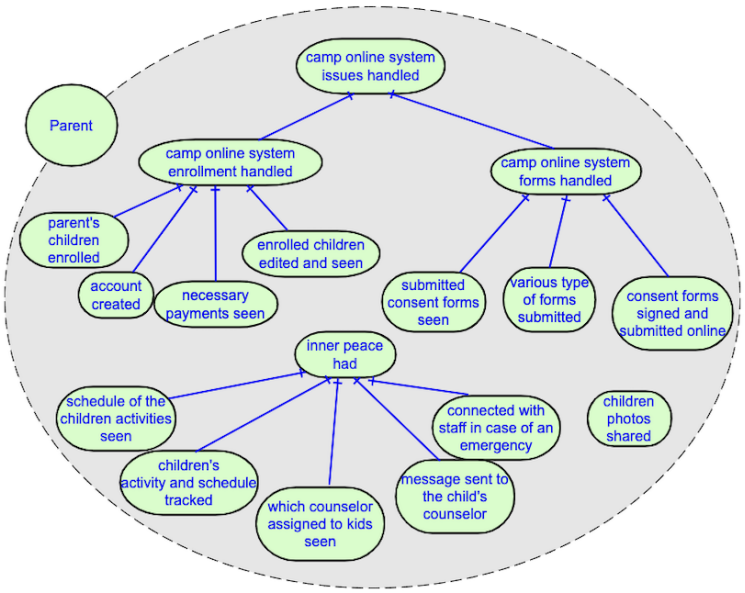

Figure 3.2. Goal model of Camperplus parent actor.

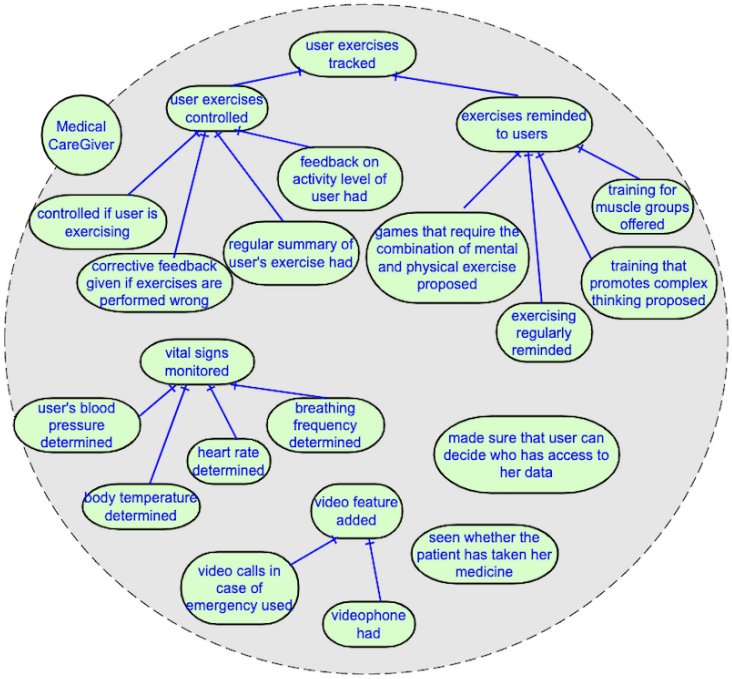

Figure 3.3. Goal model of Alfred Medical Care Giver actor.

3.2. Experiment Execution

We have 10 people as participants in this experiment. According to the participant's demographic information, 80% of them are employed as software developers in the industry, with the remaining 20% being computer science students which can be seen in Figure 3.4. Additionally, 40% of these individuals have earned their master's degree. We ask the participants to rate their level of familiarity with user stories and goal models because we are particularly interested in gauging this expertise.

Only one person in the research stated that they had never heard of user stories, showing that the industry employs them often in agile development processes. On the other hand, if we examine the data on the level of goal model experience of the participants, we can observe that 50% of them had never seen the goal models before

the experiment. Eighty per cent of our participants have never heard of the Alfred and Camperplus applications before, therefore we also want to see if they have done any prior research on the subjects throughout the experiment.

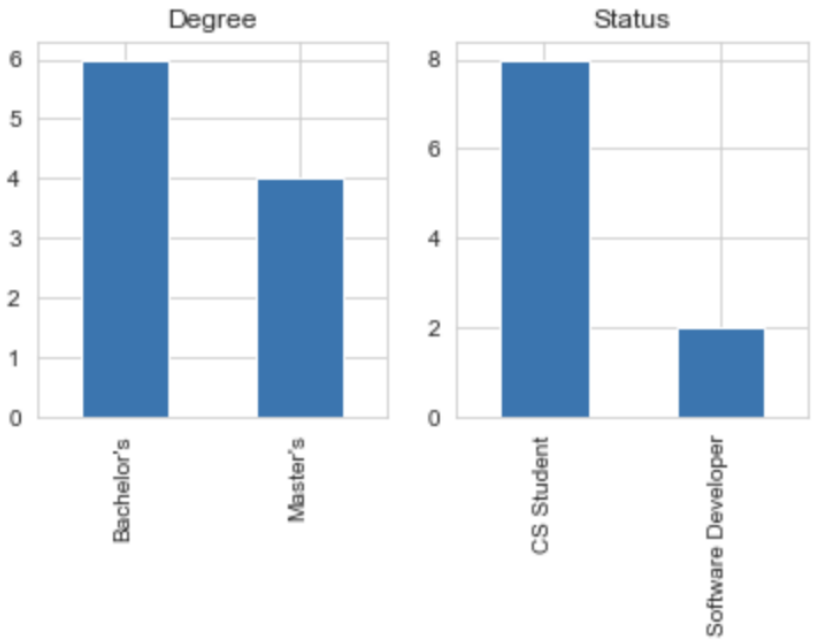

Figure 3.4. Participants' demographic results.

3.3. Results

We ask our participants to provide a general evaluation of the experiment's performance. All participants agreed that using goal models is extremely pleasant, although half of them had no prior experience with goal models. They claimed that while trying to answer questions in text format, they lost track of what they had read and had to go back and re-read it. Yet, they said that after looking at the goal model just once, the locations of the goals stayed in their visual memory and they could quickly locate them when responding to questions. In addition, it is simple to identify relationships across user stories because the goals are organized under a parent goal node.

3.3.1. Time Spent

Figure 3.5 and Figure 3.6 show the average time measurement for each question in both user story sets. Blue bars indicate the average time spent on each question given the goal model, whereas red bars are for user story set data. The bar plot makes it clear that answering questions with a goal model rather than user stories takes substantially less time.

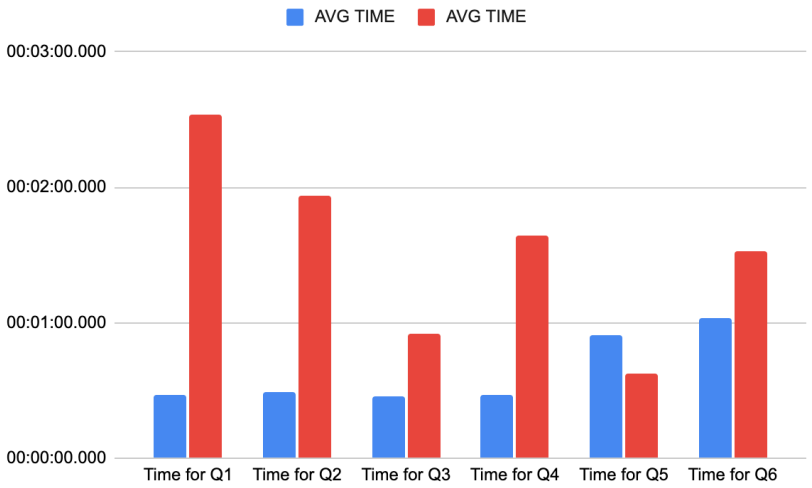

Figure 3.5. Average respond time for Alfred Data Set.

When we analyze the summary statistics on the time variable for when the goal model is given and when the user story set is given, which is shown in Table 3.4. The average time spent answering questions given the goal model is 2 minutes and 58 seconds, whereas when the user story set is given the mean value is 10 minutes and 15 seconds. Min and max time spent on answering questions about the data set are also given in the table.

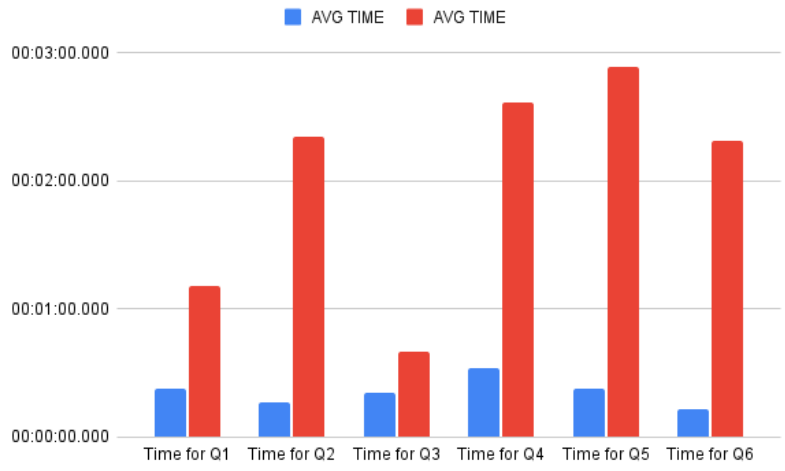

Figure 3.6. Average respond time for Camperplus Data Set.

We want to show if there is a significant difference between the time spent on answering questions by looking at the goal model or the raw user story set. To reject our null hypothesis in H_time_0 (*The time to answer data set questions from user stories and to answer from goal models are equal.*), we run statistical tests on our results.

Before choosing which statistical test to apply, we want to see if our data is normally distributed with the ShapiroWilk test. The only metric we have in this test is time in seconds. So our data have time measurements for the goal model given and the user story set given. Shapiro test results with pvalue 0.13 which we cannot say our data's distribution is normal.

Table 3.4. Statistics on time variable.

	When Model Given	**When Text Given**
Mean	00:02:58	00:10:35
Min	00:00:56	00:08:12
Max	00:05:19	00:15:54

To see whether those time differences for the goal model and user story set experiments are significantly different from each other, we follow `Wilcoxon-signed rank test` which is a non-parametric version of the two-sample t-test. The calculated p-value is 0.0019 so we can reject our null hypothesis and say our alternative hypothesis is valid which is in H_time_1 (The time to answer data set questions from user stories and to answer from goal models are not equal.)

4. GENERATING GOAL MODELS FROM USER STORIES

The first experimental results show us that goal models accelerate the process of identifying any dependencies between user stories, however creating goal models require training and time which makes them expensive. This leads to our motivation which is to provide a tool that enables users to create goal models automatically from a set of user stories.

We extract who (role), what (action, object), and benefit information from user stories using regular expressions and NLP approaches. We suggest strategies to combine this information in various ways to create goal models that resemble models created by humans. The created model is then presented in a dynamic, editable format so that human specialists can make any necessary adjustments.

Our first step in creating a goal model from a set of user stories is to utilize NLP to parse each user story. Semi-structured user stories are the starting point for NLP, which then extracts the necessary concepts and relations for our model construction. A user story's structure can be broken down into three parts. The nature of the role in the user story is described in the first part. The user story's action is stated in the second part, and its benefits are included in the third part [23].

Using regular expressions, the initial phase of our process is removing the template's special keywords from each user story. In order to divide each user story into three pieces, we used the phrases *'as a'*, *'want to'* and *'so that'* when creating the user story template. After this cleaning, part-of-speech tagging (POS) and tokenization come next. For these jobs, we employ the spaCy tool. The outcome of our pipeline's first two steps on this input is shown in Figure 4.1.

Figure 4.1. Example user story with pos tags.

Graph databases help us store extracted knowledge and make it simple to see the relationships between various components. When addressing relationships between data, graph databases are handy. In graph databases, a node and a relationship are the two building blocks that allow for the simultaneous creation of nodes and associations between them. Because `Neo4j` supports accepting many input types and has an expressive query language called *Cypher*, we decided to use it for the implementation. The fundamental Structured Query Language (SQL) principles and clauses are present in *Cypher*. In order to comprehend the relationships between the nodes and see the interconnected data, we use semantic queries to query the database when applying heuristics.

We then create fundamental heuristics to create realistic goal models. We can create numerous heuristics as our next step by using a graph database to visualize the relationships between the data. Different strategies put into practice various goal models. Making many goal models enables us to view the task from various angles. We include these strategies, and the next step will explain how to put them into practice.

4.1. Heuristics 1: Grouping Similar Verbs

This heuristic creates child nodes using the different objects related to the roles and action verbs, and parent nodes using the verbs of action of the user stories. Consider a set S that consists of three user stories (examples used here are taken from [22]):

s1: As a user, I want to be able to view a map display around my area.

s2: As a user, I want to be able to view the safe disposal events around my area.

s3: As a user, I want to view recycling centers, so that I can check which routes to take to drop off waste.

These user stories have a common role, `user`, and a verb, `view`, and each has a different object: `map display`, `safe disposal events`, and `recycling centers`, respectively (due to space limitations we use the short versions of the noun phrases). The corresponding goal model is provided in Figure 4.2.

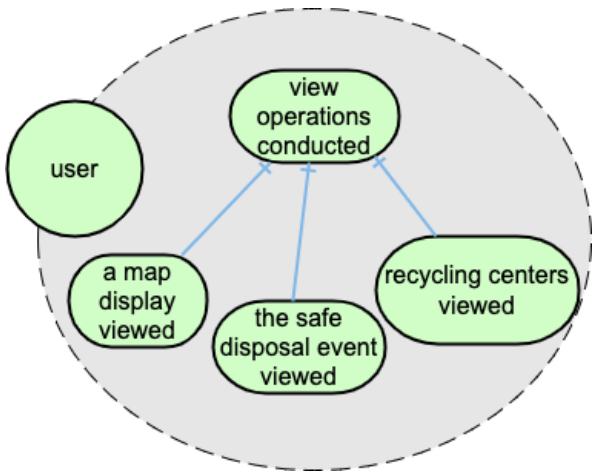

Figure 4.2. Goal model design with heuristic 1.

This procedure is described in Algorithm 4.3. It accepts as input a graph database filled with a set of user stories. The verbs of a certain role can be searched for in the graph database, which also contains the role nodes, their related verb nodes, and their respective object nodes. The technique generates an actor node in the goal model for each role in the graph database (Line 2). The verbs connected to this role are then retrieved by querying the graph database (Line 3). Then, using the similarity ratings from spaCy, it generates lists of verbs that each contain related verbs. *View* and *see*, for instance, are considered similar because their scores are higher than a particular cutoff. Line 4). The algorithm produces a parent node inside the actor boundaries for each of these lists (Line 6). Currently, we are labelling this node using the template "`verb` operations conducted". The algorithm then searches the graph database for

the objects related to the role and this specific verb for each verb in the list (Line 8). Combining the verb and the object (Line 10), makes a child node and connects it to the parent goal (Line 11). Due to space restrictions, it is not displayed in the algorithm, however, users can choose the default type of refinements as AND or OR. It provides the goal model's structure back (Line 16). In our implementation, a JSON file is returned.

```
Input a graph database USG of user stories
Output a goal model G for given user stories
for role r in USG do
  create an actor for r in G
  VL = Verbs of r
  SVL = GroupSimilarVerbs(VL)
  for list verbtree in SVL do
    Create a parent goal for the list
    for v in verbtree do
      OL is the list of objects connected to v and r
      for o in OL do
        create a child goal combining v and o in G
        link the child node with its parent in G
      end for
    end for
  end for
end for
return G
```

Figure 4.3. Grouping verbs by similarity algorithm.

4.2. Heuristics 2: Grouping Similar Objects

The second heuristic produces parent nodes from user story objects. The verbs that are connected to these parent nodes are their child nodes. Each object tree is produced inside an actor's boundaries of a role.

Consider a set S that consists of three user stories:

s1: As a user, I want to view recycling centers, so that I can check which routes to take to drop off waste.

s2: As a user, I want to know the hours of recycling facility, so that I can arrange drop-offs on my off days or during after-work hours.

These user stories have a common role, `user`, however different verbs and objects. The verbs are `view`, and `get`; and the objects are `locations of recycling centers`, and `hours of each recycling facility`. The resulting goal model is presented in Figure 4.4.

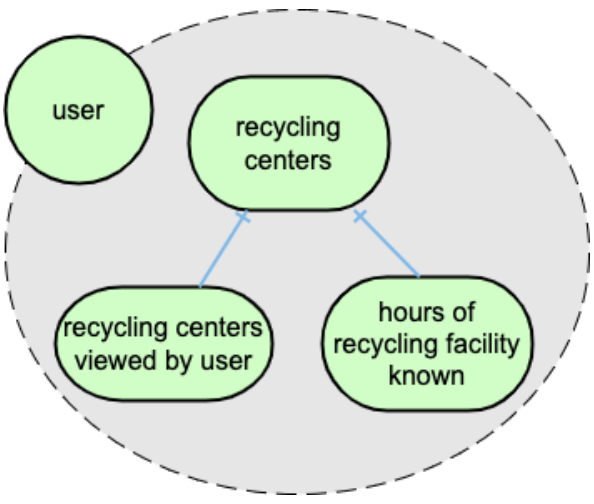

Figure 4.4. Goal model design with heuristic 2.

The transition from the collection of user stories to the goal model is described in Algorithm 4.5. Similar to Algorithm 1, it accepts as input a set of user stories that have been used to populate a graph database. The algorithm generates an actor node in the goal model for each role in the graph database (Line 2). After that, it makes a query to the graph database to retrieve the objects related to this role (Line 3). The process then generates lists of objects, each list containing related items (for phrases, we use the similarity scores from spaCy) (Line 4). The algorithm produces a parent node inside the actor boundaries for each of these lists (Line 6). For the label of this node, we now employ the template "`object` operations done." The program then searches the graph database for the verbs related to the role and this object for each object in the list (Line 8). As in Algorithm 1, it joins the verb and the object to produce a child node (Line 10) and ties it to the parent goal (Line 11). Finally, it returns the goal model's structure (Line 16).

```
Input a graph database USG of user stories
Output a goal model G for given user stories
for role r in USG do
  create an actor for r in G
  OL = objects associated with r
  SOL = GroupSimilarObjects(OL)
  for list verbtree in SOL do
    Create a parent goal for the list
    for o in objecttree do
      VL is the list of verbs connected to o and r
      for v in VL do
        create a child goal combining o and v in G
        link the child node with its parent in G
      end for
    end for
  end for
end for
return G
```

Figure 4.5. Grouping objects by similarity algorithm.

4.3. Heuristics 3: Without Actors

Consider a set S that consists of three user stories:

s1: As an executive, I want to access to company data, so that I can have a sense of my company's performance.

s2: As an employee from the HR department, I want to modify the company data.

s3: As an employee, I want to view the company data.

These user stories have the same or similar objects, `data of the company`. Their roles are different: `executive`, and `employee`. They also have different verbs, namely, `access`, `modify`, and `view`. Figure 4.6 presents the resulting goal model.

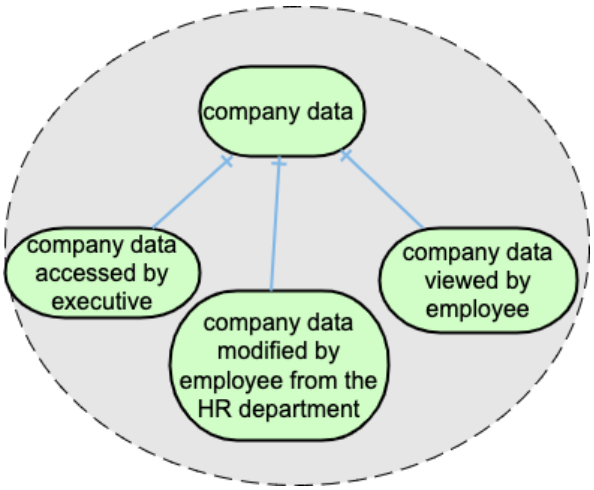

Figure 4.6. Goal model design with heuristic 3.

Heuristic 3 is similar to Heuristic 2 in that it also builds subtrees of objects, however unlike Heuristic 2, this time the goal model does not include any actor boundaries, and the child node labels also include the role information.

Beginning with Line 1 of the graph database, Algorithm 4.7 extracts the objects and groups them based on similarity (Line 2). A parent node is created for each group (Line 4). It analyses all the verbs for each element in a particular group (Line 6) and gets the roles that go along with each verb (Line 8). Each time a role and verb combination is used (Line 10), a child node is created by the algorithm and linked to the parent goal(Line 11). It concludes by returning to the goal model's structure (Line 16).

```
Input a graph database USG of user stories
Output a goal model G for given user stories
OL is the list of all objects in USG
SOL = GroupSimilarObjects(VL)
for list objecttree in SVL do
  Create a parent goal for the list
  for o in objecttree do
    VL is the list of verbs connected to o
    for v in VL do
      RL is the list of roles connected to o and v
      for r in RL do
        create a child goal combining r and v in G
        link the child node with its parent in G
      end for
    end for
  end for
end for
return G
```

Figure 4.7. Grouping objects without actors algorithm.

The output file in JSON format must then be converted into an editable visualization as the following step in the pipeline. We are currently developing a browser-based modelling editor that automatically creates a layout for the goal model utilizing the joint.js JavaScript library after parsing the JSON file containing the structural information.

4.4. Heuristics 4: Grouping Benefit of User Stories

This heuristic makes use of the benefits part of the user story. Parsing user stories by checking if a set of user stories consists of similar benefit parts allows us to see the common benefit in the user story set. This heuristic aims to see which actions lead to achieving a common benefit.

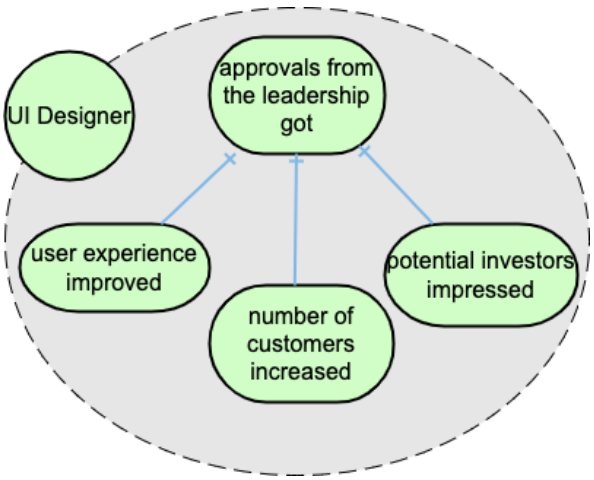

Figure 4.8. Goal model design with heuristic 4.

Here is an example of having the same benefit of 3 user stories in a set:

s1: As a UI designer, I want to improve user experience, so that I can get approvals from the leadership.

s2: As a UI designer, I want to increase the number of customers, so that I can get approvals from the leadership.

s3: As a UI designer, I want to impress potential investors, so that I can get approvals from the leadership.

For the same role type (*UI designer*), they have the same benefit which is *get approvals from the leadership* with different action objects, corresponding goal model is in Figure 4.8.

```
Input a graph database USG of user stories
Output a goal model G for given user stories
for role r in USG do
  create an actor for r in G
  BL = Benefits of r
  SBL = GroupSimilarBenefits(BL)
  for list benefittree in SBL do
    Create a parent goal for the list
    for b in benefittree do
      AL is the list of actions connected to b and r
      for a in AL do
        create a child goal taking a in G
        link the child node with its parent in G
      end for
    end for
  end for
end for
return G
```

Figure 4.9. Grouping benefits by similarity algorithm.

The extraction process of the common benefits among user stories is described in Algorithm 4.9. While consuming the set of user stories, it stores the information from the actions and benefits of user stories in a graph database. It first creates an actor node for every role in the user story set (Line 2). By calculating the similarity of the stored benefits (Line 4), for every benefit list created within a certain similarity threshold, the algorithm creates a parent node to collect them (Line 6). Later, it queries for the action part of the user story associated with the common benefit part and creates child nodes named by their actions under the benefit. These benefits are grouped in different lists now and can have their child nodes linked to the parent node.

Linking these actions to their parent nodes is processed in Line 11. The heuristic finally returns the grouped benefit structure of the goal model in Line 16.

4.5. Heuristics 5: Grouping Benefits without Role Boundary

This heuristic is a more complex one by combining heuristic 3 (getting rid of the actor) and heuristic 4 (grouping benefits) meaning that it does not require actor boundaries while trying to look for similar benefits in the set. The role information is added to the action part in the goal model which you see in 4.10. This design belongs to the below example of user stories:

s1: As a UI designer, I want to improve user experience, so that I can get approvals from the leadership.

s2: As a manager, I want to increase the profits, so that I can get approvals from the leadership.

s3: As a developer, I want to increase efficiency, so that I can get approvals from the leadership.

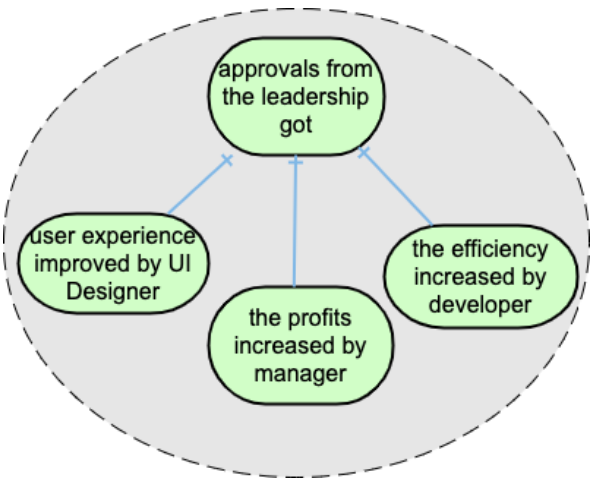

Figure 4.10. Goal model design with heuristic 5.

Beginning with Line 1 of the graph database, Algorithm 4.11 gets the action and benefit part information from user stories and creates a list of all benefits (Line 1). The similarity function collects the common benefits with their related actions (Line 2). A parent node is created for each group with similar benefits (Line 4). It analyses all the actions for each element in a particular group (Line 6) and gets the roles that go along with each action (Line 8). Each time a role and action combination is used (Line 10), a child node is created by the algorithm and linked to the parent goal (Line 11). It concludes by returning to the goal model's structure (Line 16).

```
Input a graph database USG of user stories
Output a goal model G for given user stories
BL is the list of all benefits in USG
SBL = GroupSimilarBenefits(VL)
for list benefittree in SBL do
  Create a parent goal for the list
  for b in benefittree do
    AL is the list of actions connected to b
    for a in AL do
      RL is the list of roles connected to a and b
      for r in RL do
        create a child goal combining r and a in G
        link the child node with its parent in G
      end for
    end for
  end for
end for
return G
```

Figure 4.11. Grouping benefits without actors algorithm.

5. THE GOAL MODEL BUILDER TOOL

For the industry to profit from the research on goal models, ArTu's main goals are to reduce the effort involved in creating goal models and boost goal model adoption. Simply said, ArTu takes a set of user stories, builds a goal model based on the user's chosen strategy, and offers a browser-based modelling editor so the user may update the goal model and export it in several forms. Figure 5.1 explains how it works.

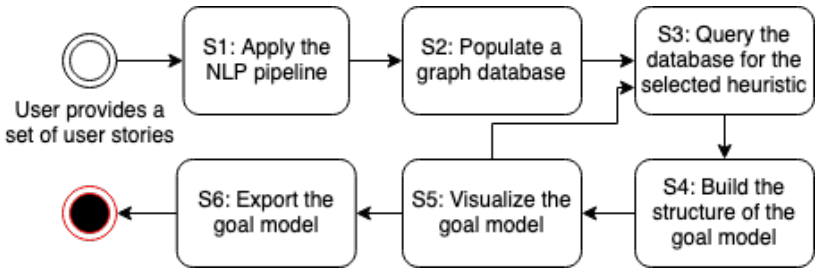

Figure 5.1. The pipeline of the process.

ArTu is a web application tool served on Heroku that produces goal models from user story sets using NLP features. Produced goal models can be viewed on an interactive editor page for users to edit and save their changes on the model.

The ArTu tool's architecture is demonstrated in Figure 5.2. The tool's user stories serve as its entry point, from which the NLP module extracts the informative phrases and stores them in the graph database (back-end module). The output JSON file is rendered for creating a goal model using React.js and the piStar library after we query the graph database (front-end module). The renderer re-connects to the NLP parser and database to perform the necessary modifications on both the backend and frontend components when modifying the final goal model.

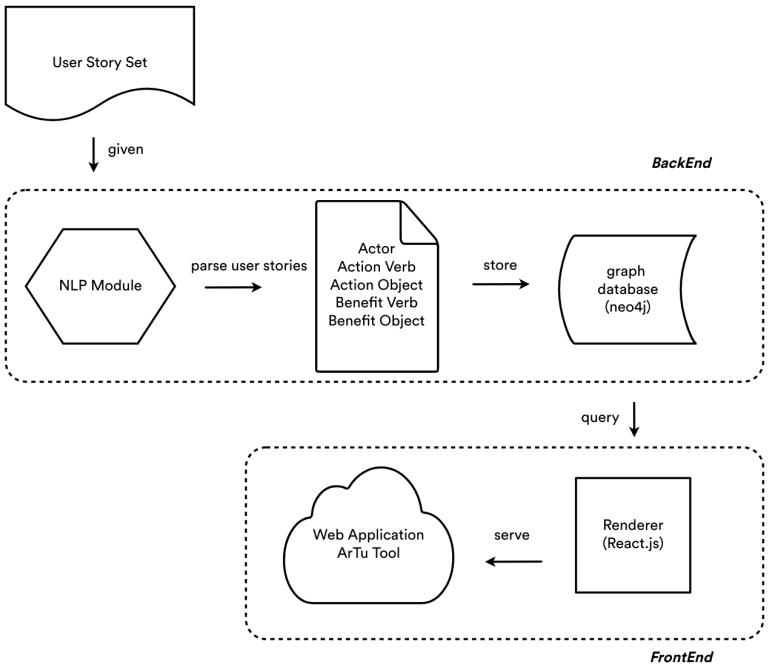

Figure 5.2. The architecture of ArTu tool.

User story sets are taken as raw text input having the following format: "As a role, I want action, so that benefit". As an example, "As a student, I want to upload my homework, so that I get good grades." The user stories are analyzed in the NLP module, written in Python, and the extracted information is stored in a graph database.

Regarding our strategies to build a goal model, we write queries on a graph database and get grouped information in JSON format. These query results are fed into our visualization module which uses React.js and piStar libraries.

While analyzing the user stories, each of them is split into 3 parts regarding their structure (role, action, benefit). After unnecessary parts are removed, user stories are tokenized and the NLP module uses Part of Speech Tagging to extract relevant information from these 3 parts. For instance, from the user story above we have *student*, *upload my homework* and *get good grades* as necessary parts. After having

the informative word phrases from user stories, they are grouped using designed 5 heuristics. To build our goal models relying on strategies, we use similarity-checking functions to group action verbs, action objects, and benefit parts by their meaning.

The ArTu tool can now create a goal model utilizing goals as nodes and relationships as edges between nodes once the similarity function has finished identifying goals and relationships. The type of the node determines how the goal nodes are labelled; for example, an actor node receives an actor label. The data collected from graph databases also label parent and child nodes, for example, the node type "goal" has information about the goal with a label and its child nodes. When connecting the nodes between the parent goal and child goal, the ArTu tool refines its lines using relationship information to generate AND or OR refinements. For instance, the example JSON in Figure 5.3 has *good grades got* as a "parent node" and *homework uploaded* as a "child nodes".

```
{
  "type": "goal",
  "label": "get good grades",
  "children": [
    {
      "type": "goal"
    },
    {
      "label": "upload my homework"
    },
    {
      "relationship": "and"
    }
  ]
}
```

Figure 5.3. Example Json format.

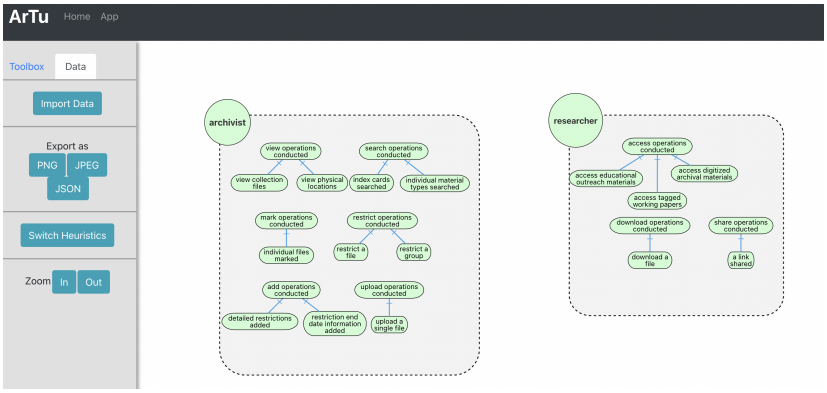

Figure 5.4. Playground page with data-tool box.

When the user sees the goal model on the tool's editor page, these are the functionalities she/he can do:

(i) Zooming in and out: Users can zoom in and out on the goal model on an interactive page, which enables them to generate an overview of a project or to concentrate on particular aspects.

(ii) Tool Bar: On the left side in Figure 5.5, the elements to draw a goal model is listed. Clicking on the element and clicking on the blank page creates the item you want. *Boundary* draw an actor boundary, *Actor* element come with an actor and its boundary, *goal* element creates goal nodes within the actor boundary. Users can edit the actor's and goal's labels by double-clicking on them. *AND* and *OR* refinements are also provided to connect nodes on the model. By default, automatically designed goal models come with AND relations, the user can easily replace it with OR by deleting the element on the model.

(iii) Export Data: Figure 5.4 shows the data features users can do on the tool. Once the user edit/create a goal model as they wish, they can export it in PNG, JPEG and JSON formats. So that they can either visualize the model or download it as JSON and work on it later on.

(iv) Import Data: If the users have their goal model data in JSON format, they can easily import them on the data toolbar and do further editing.

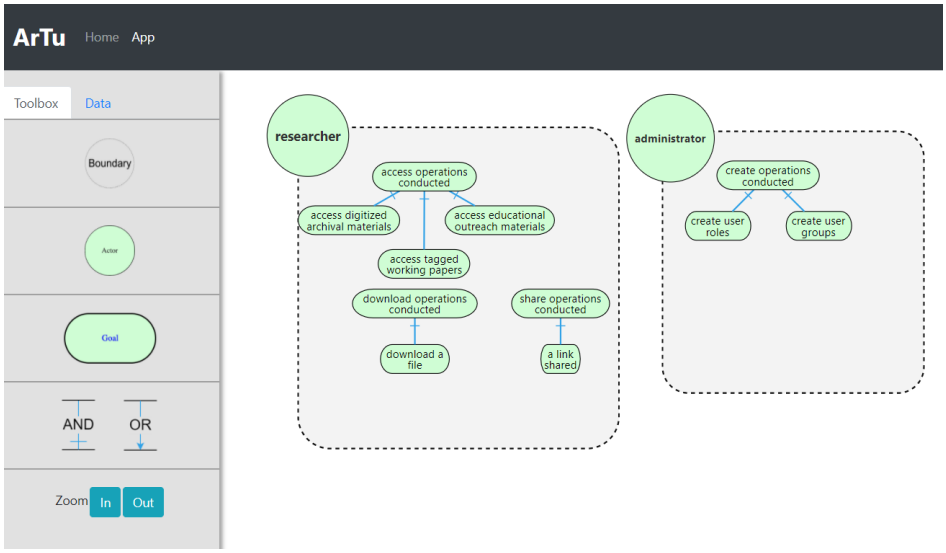

Figure 5.5. Screenshot of the modeling editor.

(v) Switch Heuristics: User stories can be uploaded, and tool heuristics can be used to create a goal model. When the tool displays it on an interactive page, the user can first examine the design before deciding whether to change the heuristic to view the other goal model design. So they can accomplish it using the switch heuristics section.

On an interactive modelling canvas modelled after the piStar tool [24], the visualized goal model is displayed for users to update. A screenshot of our modelling canvas is shown in Figure 5.5. Actor, actor boundary, goal elements, and AND/OR refinements are available in the toolbox on the left. The model can be zoomed in or out by users. By altering the chosen heuristic on the canvas, they may also test out other goal model architectures. The user can export the model as an image or JSON file after she is happy with the model. You can reach our tool in the GitHub repository [25].

ArTu tool has 3 types of users. The first group consists of professionals in the business who use agile development. They can easily input their user stories into the tool and view the goal models that are generated. The instructors who teach software engineering make up the second group of users. They can demonstrate several techniques for capturing the requirements using our technology. The third group consists of students who receive training in goal model construction. By giving them an initial goal model structure for their user story data, the tool can help them.

6. EVALUATION

The evaluation protocol is explained in this chapter. We follow the Goal Question Metric (GQM) approach [20] to build our methodology on assessing the usefulness and effectiveness of goal models and ArTu tool. To determine the quality of the designed heuristics and the tool, we study time efficiency and the complexity of the goal models. Our metric for evaluating time is time in seconds, and model complexity is the number of model elements.

We build 4 hypotheses to measure our quality metrics mentioned above. H_time_[0-1] are hypotheses to see time efficiency when using the tool to draw a goal model and when not using it. H_complexity_[2-3] are designed to evaluate the number of model elements between automatically built goal models and manually built goal models to see how complex goal models are.

Our null hypotheses:

- H_time_0: The time spent to draw a goal model without using the tool is equal to when the tool is used.
- H_complexity_2: The number of goal model elements in the tool generated one and manually generated one is equal.

Alternative hypotheses state:

- H_time_1: The time spent to draw a goal model without using the tool is longer than when the tool is used.
- H_complexity_3: The number of goal model elements between tool-generated one and manually generated one is not equal.

6.1. Experimental Design

Purpose. Here we aim to design an evaluation methodology to test our hypotheses above. The time difference to build a goal model between using the tool and without using it is one of our quality issues. To measure time, *time in seconds* is used as a metric. The model elements are considered to indicate the complexity quality when drawing goal models. Each participant is asked to send their goal model design after the experiment ended. We analyze those designs to calculate the number of model elements as a metric for this quality.

Experiment Protocol. In this part, we run a cross-over experiment on participants with 2 user story sets. Each participant who gave consent to participate is assigned to two different groups (treatment group and control group). This experiment includes 2 sessions. During the first session, the control group receives one of the user-story sets and draws a goal model manually using ArTu tool's playground page, while the treatment group receives the other user story set and uses built heuristics to have an automatically generated goal model. Then during the second session, the order of having goal models either manually or automatically is reversed according to the crossover experiment design. In other words, the control group receives the second user story set and uses the tool to draw automatically. Whereas the treatment group receives the first user story set and draws manually on the tool. Table 6.1 shows how we set up the cross-over design in this experiment.

Table 6.1. Experiment 2 setup design.

	With Tool	**Without Tool**
User Story Set 1	Group 1	Group 2
User Story Set 2	Group 2	Group 1

Before the experiment, all participants get an e-mail including an explained tutorial, 2 user story sets, and ArTu tool link. Moreover, a short questionnaire form is sent to gather information about participants' profiles. Participants in the online experiment were asked to share their screens during the session. Each participant is asked to provide feedback regarding the experiment's progress and the session they found easiest, both verbally and in writing on a questionnaire, at the end of the study. The flow of this experiment is shown in Figure 6.1.

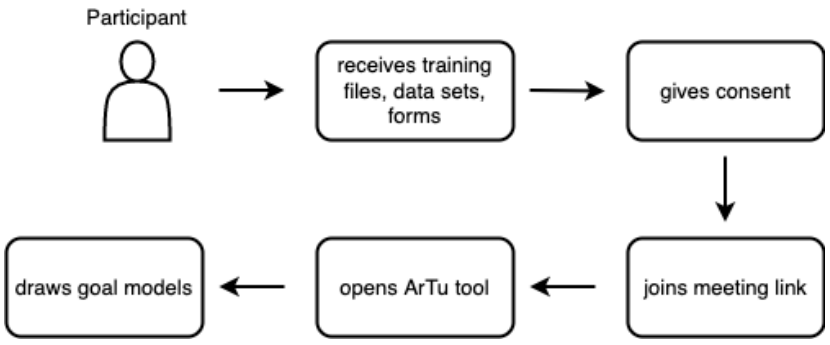

Figure 6.1. Experiment 2 flow.

Materials. Before the experiment, each participant receives a tutorial that includes the same user stories and goal model information as in the first experiment. In addition, it includes instructions on how to use the ArTu tool to upload a set, choose heuristics, and draw a goal model on a blank playground page using tool elements. For this experiment, two user story sets are used in two separate sessions. To learn more about our users' profiles and how they feel about the usability of the ArTu tool, a feedback form in Table 6.3 and a demographics form in Table 6.2 are also delivered. Participants are required to download goal model designs from the tool at the end of each session, thus we had 2 goal models.

Table 6.2. Demographics form questions & answers.

Demographics Questions	**Answers**
Which of the following categories best describes your employment status?	Computer Engineering Student Software Developer Not Employed Researcher Other
What describes you if you are still a student?	Bachelor's degree Master's degree PhD or higher
Have you taken any training on user stories?	No I've taken a course at university I've taken a training at work Other
How much experience do you have in User Stories?	No experience I use them at work I'm actively writing user stories
Have you taken any training on goal models?	No I've taken a course at university I've taken a training at work Other
How much experience do you have in Goal Models?	No experience I use them at work I'm actively writing user stories

Table 6.3. Feedback form questions & answers.

Feedback Questions	**Answers**
Depending on your role (teacher, student, worker) would you use it for your work purposes?	Yes, because ... No, because ... Maybe, because ...
How easy is it to edit the automatically built goal model?	Linear scale from 1 to 5 1 Very easy 5 Complex
Do you prefer editing the automatically drawn goal model to have your goal model or building your model from scratch?	Yes No
What would you rate the user-friendliness of the tool?	Linear scale from 1 to 10 1 Very easy to navigate 10 Complex

6.2. Experiment Execution

The experiment involves computer engineering students, software developers and researchers. From 38 participants 25 of them are software developers which are shown in Figure 6.2.

92% of the participants have studied user stories, and 79% of them have studied goal models. These results show that most of them have preliminary knowledge of user stories and goal models which you can see in Figure 6.3 and Figure 6.4, also 80% of them studying for their master's degree in software engineering.

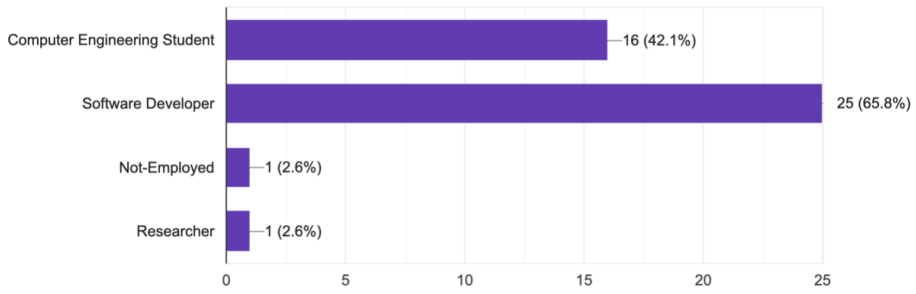

Figure 6.2. Participants' demographic results.

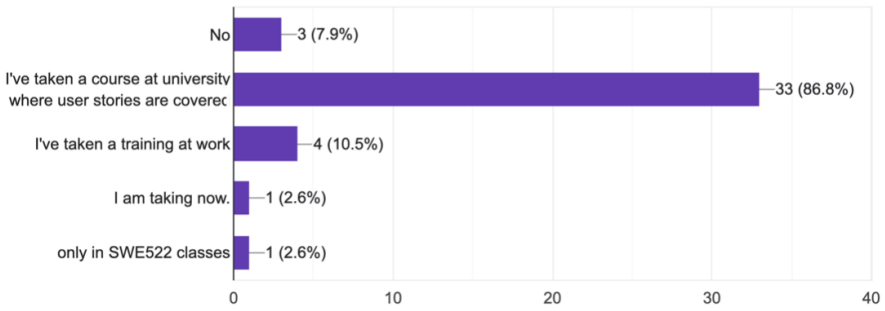

Figure 6.3. Participants' user story knowledge.

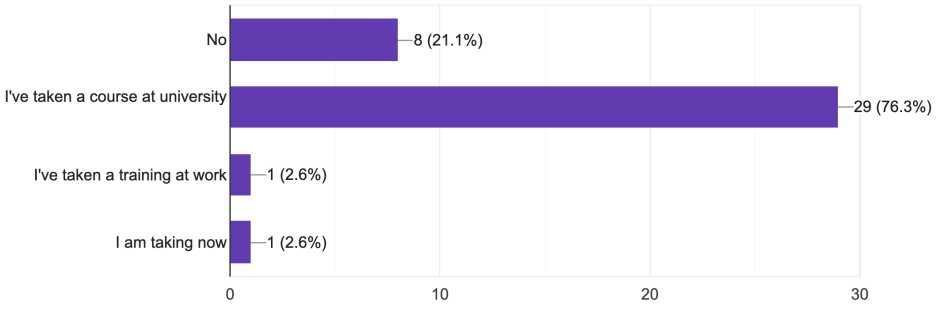

Figure 6.4. Participants' goal model knowledge.

However, a demographics survey reveals that nearly half of them have no experience with either user stories or goal models in the workplace. 30% of them say they do not write user stories but instead use them for work projects. And rest 20% of users actively write user stories. On the other hand, 65% of the participants lack any expertise with goal models rather than taking a course in university. 28% use goal models at their job but do not draw practically, and only 7% of them draw/use goal models actively at work.

6.3. Results

In this section, we first discuss the similarity between the 2 user story sets by showing the time spent by each participant. Then we examine the time difference to draw goal models with the tool and without the tool and lastly, goal model elements are considered.

6.3.1. User Story Similarity

Our first aim is to show that 2 user story sets used in the experiment have similar complexity and to achieve that we run some statistical analysis on the data. Before deciding which statistical test to run, we analyze whether the data we collected from measuring time for these 2 data sets have a normal distribution. *Sharpio-Wilk* test is performed on the data, and the p-value is greater than 0.05 which indicates we have normally distributed data.

In Figure 6.5, the time spent in seconds using the tool and not using the tool is shown. When we first look at grouped box plots for both data sets, the average time spent when the tool is not used is very similar which means our data sets are similar complexity-wise.

Figure 6.5. Boxplot for 2 datasets showing time spent.

An independent t-test is chosen to prove that there is no significant difference in time spent on both user story sets. Total time which indicates time spent with the tool and without the tool is analysed for both sets with a t-test, the result for the p-value is equal to 0.575. So we can say that these 2 sets are not different from each other.

6.3.2. Time Difference

After showing the similarity between the two user story sets, our plan is to show if there is a significant time difference between drawing a goal model with the tool and without the tool. H_time_[0-1] hypotheses are tested in this part for control and treatment groups. Figure 6.6 displays the distribution of time in seconds for Archive data set for when the tool is used and when is not used. It is ame for Data Management data set is shown in Figure 6.7. We can easily see that for both data sets the average time spent in seconds when the ArTu tool is not used is higher than the tool used.

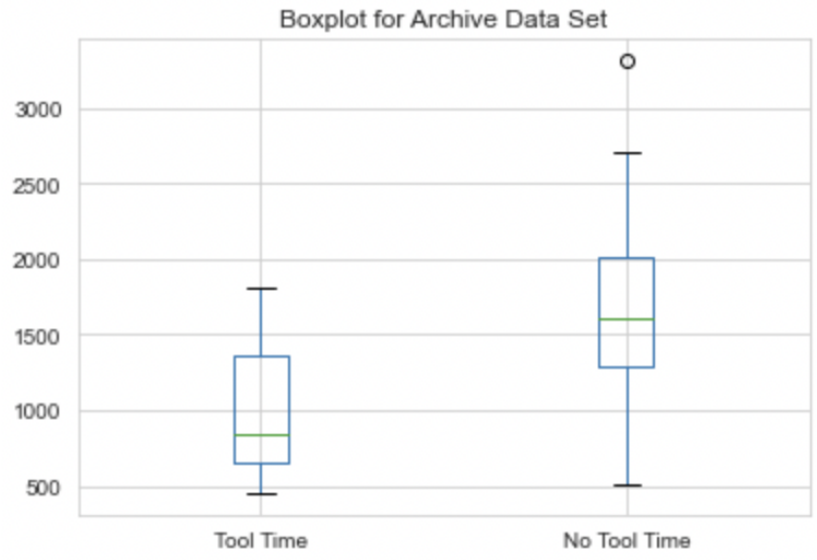

Figure 6.6. Boxplot for archive set showing time in seconds.

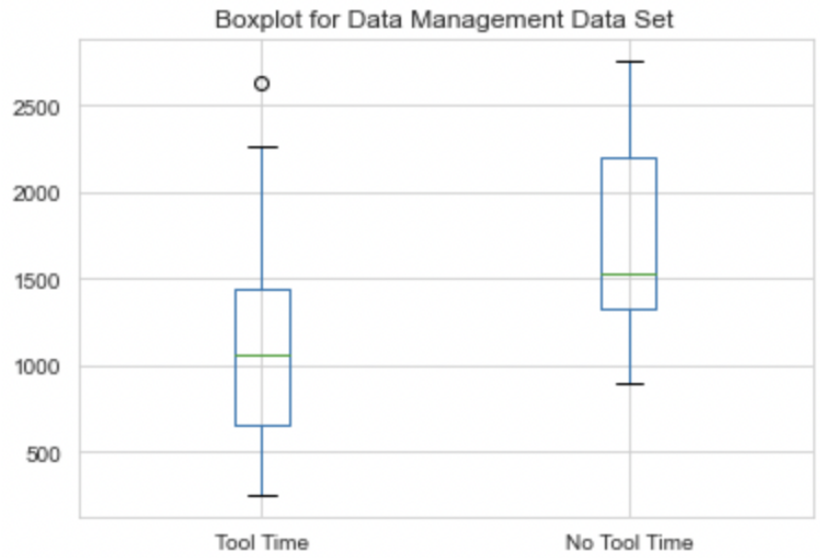

Figure 6.7. Boxplot for data management set showing time in seconds.

When we analyse the summary statistics on the time variable for when the tool is used and when not which is shown in Table 6.4. The mean value for tool time is 17 minutes, while the mean value for a tool not used is 28 minutes.

This time we apply Shapiro-Wilk test on each data set for tool times and no-tool times to see if they have normal distributions. Both user story sets have normally distributed data when we look at time in seconds (both p-values are greater than 0.05). So the test is not significant which indicates the sampling distribution is normally distributed. To see if these time values are significantly different from each other we use `independent t-test` considering the data we have. The result from the t-test indicates there is a significant difference in time when the tool is used to draw.

Table 6.4. Statistics on time variable.

	Time Spent With Tool	**Time Spent Without Tool**
Mean	00:17:45	00:28:09
STD	00:09:00	00:10:15
Min	00:04:12	00:08:34
Max	00:43:51	00:55:10

The pvalue is equal to 0.002 which we can reject our null hypothesis.

- H_time_0: The time spent to draw a goal model without using the tool is equal to when the tool is used.
- H_time_1: The time spent to draw a goal model without using the tool is longer than when the tool is used.

6.3.3. Goal Model Elements

Our second and third hypotheses try to show if there is a difference in terms of the number of goal model elements between tool-designed and manually-designed goal models.

- H_complexity_2: The number of goal model elements in the tool-generated one and manually generated one is equal.
- H_complexity_3: The number of goal model elements between the tool-generated one and manually generated one is not equal.

First, we analyse the data to see if it has normal distribution by using the Shapiro-Wilk test. When it has been applied, the result has shown we have normally distributed data. Thereby we apply a t-test in this case as well. Since we have a p-value of 0.316, we cannot say they are significantly different.

However, when we look at the distributions of model elements for both sets, we see a consistent number of elements when the model is drawn with the tool. On the other hand, the number of manually drawn goal model elements is spread over the figures. This leads us to conclude having a goal model structure using the tool gives a consistent design of the goal model. The distribution for Archive Set can be seen in Figure 6.8, while Data Management set in Figure 6.7.

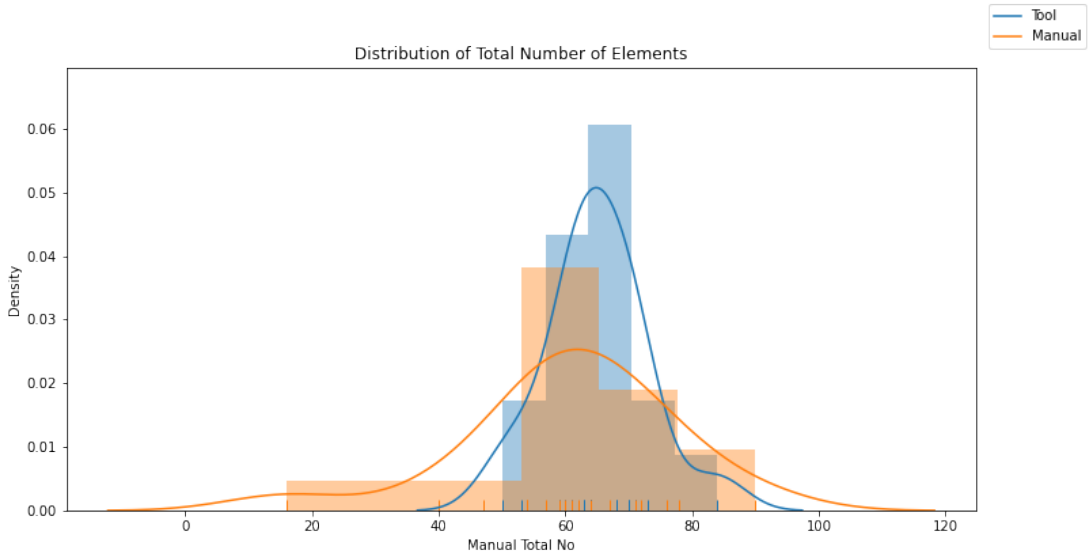

Figure 6.8. Distribution of number of goal models elements for Archive Data Set.

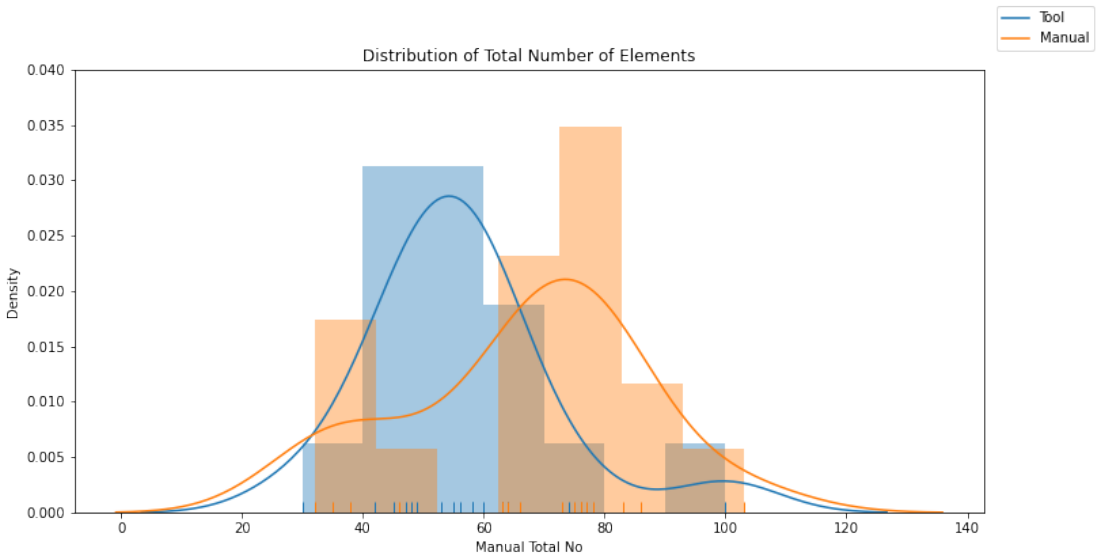

Figure 6.9. Distribution of goal models drawn for Data Management Data Set.

7. THREATS TO VALIDITY

In terms of external threats, internal threats, construct threats, and reliability threats, we analyze the limitations of our solution in this chapter. The threats to the validity considered in this study are explained in the next sections.

7.1. External Validity

Threats to external validity decrease the results' generalization. *Experience:* It is unlikely that participants with similar levels of expertise in user stories and goal models will participate in this study. To minimize this threat, we ask people from a software engineering background. *Number of participants:* More participants could join this study to strengthen the obtained results, yet we have 38 participants and a cross-over design which lessens the threat. *Varieties of user story sets:* Evaluations on additional data sets are necessary to obtain more trustworthy accuracy results. Yet, the results have been proven to be good.

7.2. Internal Validity

Threats to the internal validity of the centre of the experiment on their methodology. *Data sets:* The data sets were selected from a public place which we did not write ourselves so we limit the threat here. *Time:* In the study there are 2 sessions, and 30 minutes is assigned for each participant, but they are also allowed to take their time in each session. *Heuristics:* To have a goal model automatically, the strategies are described in detail before the experiment. Users choose the heuristic as they want, yet they mostly select the heuristic 1. *Learning:* Users who started using the ArTu tool with heuristics have learned how to draw a goal model from their first session. *Maturation:* It is possible that participants lost motivation or performed worse due to fatigue after completing the first session. These threats are reduced by having a cross-over designed experiment.

7.3. Construct Validity

Threats related to how well a test captures what it is designed to capture. While building goal models on the tool, the designs do not have OR refinement however users can adjust the goal model design as they see fit. So this threat is dropped by allowing users to edit the goal models as they wish.

7.4. Reliability

In terms of reliability, we have four hypotheses covering the number of goal model elements and the time spent when drawing goal models. We have carried out statistical analyses and the results demonstrated the validity of our hypotheses.

8. DISCUSSION

In this chapter, we analyzed the results we got from people at the end of their experiment. Moreover, we go over the comments made by users and read the feedback survey form. At the end of each session, we encouraged them to discuss the tool's usability, the clarity of its heuristics, and what could be done to make the tool and the experiment better.

8.1. Heuristics

Each participant receives a tutorial describing the heuristics on ArTu tool before their experiment. During the experiment, they are asked to revise the user story sets and choose which heuristic design to utilize when drawing a goal model. The selected heuristic numbers are saved for every user, and heuristic 1, which groups user stories with similar action verbs, was picked by 28 participants. This might be a result of the comparable structure of both user story sets. Heuristics 2,4, and 5 are used seven times, four times, and once respectively.

8.2. Feedback Questionnaire

When asked if they would use the tool at work, more than half of the respondents answered they would, with 11% saying they may. Eighty per cent of users said it was very simple to update the goal model once they had it from the heuristics. When asked if they would use the tool to generate goal models generally, 85% of them replied yes, and 90% stated they preferred automatically constructed goal models to ones that were manually drawn. 95% of users give the tool a rating of 7 or higher due to its user-friendliness.

8.3. Overall

Overall feedback covers what the participants said after their experiment ended. The participants who started using the tool to generate goal models indicated that in their next session (drawing manually) they learned how to draw from automatically drawn one and felt biased when they had to draw by themselves. Some of them even created their goal model mimicking the heuristics design. According to our overall analysis of time differences, it can be seen that using the tool speeds up the process, yet few participants said they felt more at ease when they had to draw manually.

Heuristics were unclear to some participants, and they needed extra time to practice drawing goal models with heuristics. Some of them desired further illustrations of what the heuristics did or the situations in which they should apply whichever heuristic. Goals are linked to one another via AND refinements in all of the heuristic-generated goal models. But several people claim to have used OR in some cases, thus it should also be added. People who utilize heuristic 4 advise using the benefit component with a quality attribute in iStar framework.

Both sets of user story sets, which each have 27 user stories, are thought to be comprehensible and contain a good number of user stories. Users claim that even though the set size was relatively tiny for a software project when they had goal models created from scratch, some user stories were missing in the manually drawn goal model. On the other hand, they also claimed that the automatically generated goal models always include user stories, making them useful. They claimed that if the user story sets were larger, the time gap would be significantly greater. When asked which method was simpler, several users who spent about the same amount of time drawing with and without the tool chose to modify the automatically created model. Most of them claimed that starting with the heuristic-developed goal model made a significant difference in terms of easy editing and requiring less time.

9. RELATED WORK

An unexplored field of research is automated model development from text documents using NLP. For instance, Sanyal and Kumar [26] provide the SUGAR tool, which uses NLP and syntactic principles to construct use cases and class diagrams from NL requirements. Using difficult NL requirements, Deeptimahanti and Babar [27] present a method for producing UML models. The authors describe a tool called UML Model Generator from Analysis of Requirements (UMGAR), which can handle lengthy requirements documents and is powered by NLP technologies like Stanford Parser [28] and WordNet [29].

Using NLP and domain ontology approaches, More and Phalnikar [30] create the RAPID tool to extract UML diagrams from NL specifications. They intend to automate the conversion of textual user requirements to UML class diagrams using NLP and domain ontology, which is similar to earlier work by Herchi and Abdesselam [31]. In order to capture notions like class names, their characteristics, and linkages to create UML, the technique finally uses some linguistic rules.

Ibrahim and Ahmad [32] utilise the RACE tool, which is very similar to the RAPID tool in that it combines NLP and domain ontology knowledge, to translate user requirements specified in NL into class diagrams. By layering conceptual patterns on top of NLP methods, Letsholo et al. [33] offer the TRAM tool for automatically creating analytical models from natural language requirements. Additionally, the system enables users to get in touch with experienced modellers to raise the model's quality. Abdessalem et al. [34] suggest an automated program that will generate UML class diagrams from NL requirements by extracting the class elements using NLP techniques like pattern matching.

Researchers are becoming more interested in user stories as agile development approaches to gain traction in the software industry. Using the natural language processor

tool spaCy, Robeer et al. [35] present a completely automated solution to generate a conceptual model from user stories. In a further piece of work, Lucassen et al. [36] suggest the Visual Narrator tool, which extracts conceptual models from user tales using Python and the Natural Language Tool Kit (NLTK). Elallaoui et al. [37] suggest employing NLP techniques to automate the conversion of user stories into UML use case diagrams. In contrast to other studies, this one evaluates user stories while creating UML diagrams, whereas other studies mostly focus on requirements documents. Kochbati et al. [38] generates UML models from user stories. To divide the system into smaller units, they create a semantic similarity module that groups the natural language requirements. Gulle et al. [39] cluster topics within crowd-generated user stories using pre-trained word embeddings and Word Mover's Distance (WMD). Resketi et al. [40] attempt to employ NLP features to summarize a set of user stories based on their frequency and then reuse them in future projects of a similar nature.

Elallaoui et al. [41] develop an algorithm that converts XML files from user story-contained files into sequence diagrams by utilising the UML2 tool SDK plugin for Eclipse. By creating a Visual Narrator tool to extract conceptual models from user stories, Lucassen et al. [42] study potential remedies for using algorithms that have a semantic understanding between ideas to display user stories. Arora et al. [43] suggest using NLP tools to automate the process of building domain models from NL requirements while also providing additional rules using the NLP dependency parser.

The conversion of user stories to goal models has received some attention. To automate the creation of iStar models from user stories, [44] create the tool US2StarTool. Lin et al. [45] present a method for modelling goal requirements from user stories that are goal-oriented. The approach Goal Net shows goal structure as ranging from simple user stories to complex objectives. For user story sets, Wautelet et al. [23,46–48] create rationale models. These papers focus on breaking down user stories into three broad pieces rather than analyzing the smaller linguistic components within them (actor, action, benefit). Our research differs from theirs in that we want to be more specific and do not create goals for the entire action or benefit portion of a user story instead

we select objects and verbs and connect them. Wu et al. [49] study generating goal models from user stories, which are built using iStar framework. They merge nodes in the model using BERT to identify node similarity.

Conducting a controlled experiment to evaluate this project's characteristics is another crucial step. When compared to not using the technology, Winkler et al. [50] want to know if it improves categorization quality. Two groups were utilized in their controlled experiment: a treatment group and a control group. One group used the tool, the other group did not. To determine which situations employing the tool is advantageous is the goal. Ko et al. [51] created a book to instruct readers on how to conduct controlled experiments. They say it's crucial for experiments to have a research question. Additionally, Abrahão et al. [52] propose a controlled experiment for contrasting the efficacy of several goal model languages.

Our evaluation strategy compares automatically generated models with human-built models by analyzing their similarities in order to provide realistic goal models. A current area of investigation is how to quantify graph similarity. A method for comparing labelled graphs that involve establishing a minimum similarity criterion is presented by Raymond et al. [53]. Utilizing the maximum common edge subgraph (MCES) discovery approach, the similarity is evaluated. When the node correspondence is known, Koutra et al. [54] suggest DELTACON to measure the connectedness between two graphs. This method addresses the issue and measures the overlap of graph edges. A novel framework for calculating the degree of graph similarity utilizing belief propagation and related concepts is designed by Koutra et al. [55] in another piece of work. They operate on two graphs with the same number of nodes but various edges. According to Zheng et al. [56], it is inefficient to gauge graph similarity across huge graph databases. In order to solve the edit-distance-based similarity problem, they propose to retrieve graphs if they are similar to the specified query graph.

10. CONCLUSIONS

It is difficult to see the relationships in user stories because they are presented with flat data. Goal models, on the other hand, give a high-level representation of the relationships and pinpoint user stories that discuss the same concepts. However, manually creating goal models requires time, effort, and training, so they are not widely adopted in the industry. To enhance the usage of goal models in agile practices, we introduce ArTu, a goal model-building tool driven by NLP.

The initial experiment on evaluating the utility of goal models motivated the development of the ArTu tool. The goal model allows users to uncover user story relationships more quickly, according to the results. To meet the problem of creating a goal model, we develop several heuristics that combine user stories and generate several goal model designs that resemble goal models created by humans. To extract useful information from each user story in the set, we apply NLP features. We then store the information in graph databases to display the dependencies between nodes and edges more clearly.

In order to display the relationships between nodes in the front-end part, we put the strategies we obtained by querying the graph database into a JSON format. The user story's action element is the emphasis of the first three heuristics, while the fourth and fifth heuristics combine the benefit and action parts. We can upload a set of user stories to ArTu, and based on the goal model structure method we choose, the goal model is generated. The model is displayed on its interactive playground page, which allows users to modify the tool-built goal model as they see fit. A goal model's final iteration is available for download as an image or in JSON format.

The purpose of the second experiment is to evaluate the effectiveness of heuristics and the ArTu tool. We have designed it to explore the metrics for time effectiveness and the number of model elements. In order to cover these quality criteria, we have

4 hypotheses. Hypotheses H_0 and H_1 focus on the time efficiency metric and ask whether using the tool may speed up the process of creating goal models. H_2 and H_3 are used to assess the model's components and determine whether the number is consistent. Other elements that need to be vocally assessed include the tool's usability and perceived usefulness.

Future research is now possible thanks to this study of the ArTu tool's and goal models' efficacy, which highlighted the significance of goal models in software development processes. In the long run, it would be interesting to use machine learning-based techniques in place of our rule-based heuristics to generate the goal models. We can enhance the amount of heuristics on the tool by combining alternative grouping techniques or working on designed goal models utilizing node similarity methodologies since feedback pointed to the need for more complicated heuristics. In addition, the playground page and toolbox features, for example, have received input regarding how they might be improved and made more user-friendly.

REFERENCES

1. Kassab, M., C. Neill and P. Laplante, "State of Practice in Requirements Engineering: Contemporary Data", *Innovations in Systems and Software Engineering*, Vol. 10, pp. 235–241, 2014.

2. Kassab, M., "The Changing Landscape of Requirements Engineering Practices Over The Past Decade", *Fifth International Workshop On Empirical Requirements Engineering (EmpiRE), Ottawa, Canada*, pp. 1–8, IEEE, 2015.

3. Lucassen, G., F. Dalpiaz, J. M. E. M. v. d. Werf and S. Brinkkemper, "The Use and Effectiveness of User Stories in Practice", *Requirements Engineering: Foundation for Software Quality: 22nd International Working Conference, REFSQ, Gothenburg, Sweden*, pp. 205–222, 2016.

4. Cohn, M., "User Stories Applied: For Agile Software Development", Addison-Wesley Professional, Redwood City, CA, USA, 2004.

5. Horkoff, J., F. B. Aydemir, E. Cardoso, T. Li, A. Mat'e, E. Paja, M. Salnitri, L. Piras, J. Mylopoulos and P. Giorgini, "Goal-Oriented Requirements Engineering: An Extended Systematic Mapping Study", *Requirements Engineering*, Vol. 24, No. 2, pp. 133–160, 2019.

6. Aydemir, F. B., F. Dalpiaz, S. Brinkkemper, P. Giorgini and J. Mylopoulos, "The Next Release Problem Revisited: A New Avenue for Goal Models", *IEEE 26th International Requirements Engineering Conference (RE), Banff, AB, Canada*, pp. 5–16, IEEE, 2018.

7. Dalpiaz, F., X. Franch and J. Horkoff, "iStar 2.0 Language Guide", *arXiv preprint arXiv:1605.07767*, 2016.

8. Nguyen, C. M., R. Sebastiani, P. Giorgini and J. Mylopoulos, "Multi-objective Reasoning With Constrained Goal Models", *Requirements Engineering*, Vol. 23, No. 2, pp. 189–225, 2018.

9. Güneş, T. and F. B. Aydemir, "Automated Goal Model Extraction From User Stories Using NLP", *IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland*, pp. 382–387, 2020.

10. Güneş, T., C. A. Öz and F. B. Aydemir, "ArTu: A Tool for Generating Goal Models From User Stories", *IEEE 29th International Requirements Engineering Conference (RE), Notre Dame, IN, USA*, pp. 436–437, 2021.

11. I. K. Raharjana, D. S. and C. Fatichah, "User Stories and Natural Language Processing: A Systematic Literature Review", *IEEE Access*, Vol. 9, pp. 53811–53826, 2021.

12. Dalpiaz, F. and S. Brinkkemper, "Agile Requirements Engineering With User Stories", *IEEE 26th International Requirements Engineering Conference (RE), Banff, AB, Canada*, pp. 506–507, 2018.

13. Amna, A. R. and G. Poels, "Ambiguity in User Stories: A Systematic Literature Review", *Information and Software Technology*, Vol. 145, p. 106824, 2022.

14. Zeaaraoui, A., Z. Bougroun, M. G. Belkasmi and T. Bouchentouf, "User Stories Template for Object-Oriented Applications", *Third International Conference on Innovative Computing Technology (INTECH), London, UK*, pp. 407–410, 2013.

15. Lombriser, P., F. Dalpiaz, G. Lucassen and S. Brinkkemper, "Gamified Requirements Engineering: Model and Experimentation", *Requirements Engineering: Foundation for Software Quality*, pp. 171–187, Springer International Publishing, 2016.

16. Kavakli, E. and P. Loucopoulos, "Goal Modeling in Requirements Engineering:

Analysis and Critique of Current Methods", *Information Modeling Methods And Methodologies: Advanced Topics in Database Research*, pp. 102–124, IGI Global, 2005.

17. Horkoff, J. and E. Yu, "Analyzing Goal Models: Different Approaches and How to Choose Among Them", *Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, Taiwan*, SAC '11, pp. 675–682, 2011.

18. Ali, R., F. Dalpiaz and P. Giorgini, "A Goal-Based Framework for Contextual Requirements Modeling and Analysis", *Requirements Engineering*, Vol. 15, pp. 439–458, 2010.

19. Horkoff, A. F. C. E. e. a., J., "Goal-Oriented Requirements Engineering: An Extended Systematic Mapping Study", *Requirements Engineering*, Vol. 24, pp. 133–160, 2019.

20. Caldiera, V. R. B. G. and H. D. Rombach, "The Goal Question Metric Approach", pp. 528–532, 1994.

21. Horkoff, J., "iStar 2.0 Core Language", `http://istar.rwth-aachen.de/tiki-index.php?page=Visio+Template`.

22. Dalpiaz, F., "Requirements Data Sets (User Stories)", Mendeley Data, 2018, accessed on June 16, 2021.

23. Wautelet, Y., S. Heng, M. Kolp and I. Mirbel, "Unifying and Extending User Story Models", *26th International Conference on Advanced Information Systems Engineering (Caise), Thessaloniki, Greece*, Vol. 8484, pp. 211–225, CAiSE, 2014.

24. Pimentel, J. and J. Castro, "piStar Tool: A Pluggable Online Tool for Goal Modeling", *IEEE 26th International Requirements Engineering Conference (RE), Banff, AB, Canada*, pp. 498–499, IEEE, 2018.

25. Güneş, T., "tugcegns/goal-model-builder: Goal Model Builder Tool", Zenodo, 2022, `https://zenodo.org/record/7011245`, accessed on September 2, 2022.

26. Deeptimahanti, D. K. and R. Sanyal, "Static UML Model Generator From Analysis of Requirements (SUGAR)", *In Advanced Software Engineering and Its Applications (ASEA)*, 2008.

27. Deeptimahanti, D. K. and M. A. Babar, "An Automated Tool for Generating UML Models From Natural Language Requirements", *Automated Software Engineering*, pp. 680–682, 2009.

28. Klein, D. and C. D. Manning, "Stanford Parser", `https://nlp.stanford.edu/software/lex-parser.shtml`, accessed on August 21, 2021.

29. Fellbaum, C., "WordNet A Lexical Database for English", `https://wordnet.princeton.edu/`, 1998, accessed on August 21, 2021.

30. More, P. R. and R. Phalnikar, "Generating UML Diagrams From Natural Language Specifications", *International Journal of Applied Information Systems*, Vol. 1, pp. 19–23, 2012.

31. Herchi, H. and W. Abdessalem, "From User Requirements to Uml Class Diagram", *arXiv preprint arXiv:1211.0713*, 2012.

32. Ibrahim, M. and R. Ahmad, "Class Diagram Extraction From Textual Requirements Using NLP Techniques", *Second International Conference on Computer Research and Development, NW Washington, DC, United States*, pp. 200–204, IEEE, 2010.

33. Letsholo, K. J., L. Zhao and E.-V. Chioasca, "TRAM: A tool for Transforming Textual Requirements Into Analysis Models", *28th IEEE/ACM International Conference on Automated Software Engineering (ASE), Silicon Valley, CA, USA*, pp. 738–741, 2013.

34. Abdessalem, W., Z.B.Azzouz, A.Singh, N.Dey, A.S.Ashour and H.B.Ghezala, "Automatic Builder of Class Diagram(ABCD): An Application of UML Generation From Functional Requirements", *Software Practice and Experience*, Vol. 46, 2016.

35. Robeer, M., G. Lucassen, J. M. E. M. van der Werf, F. Dalpiaz and S. Brinkkemper, "Automated Extraction of Conceptual Models From User Stories via NLP", *IEEE 24th International Requirements Engineering Conference (RE), Beijing, China*, pp. 196–205, 2016.

36. Lucassen, G., M. Robeer, F. Dalpiaz, J. M. E. M. van der Werf and S. Brinkkemper, "Extracting Conceptual Models From User Stories With Visual Narrator", *Requirements Engineering*, Vol. 22, pp. 339–358, 2017.

37. Elallaoui, M., K. Nafil and R. Touahni, "Automatic Transformation of User Stories Into UML Use Case Diagrams Using NLP Techniques", *Procedia Computer Science*, Vol. 130, pp. 42–49, 2018.

38. Kochbati, T., S. Li, S. Gérard and C. Mraidha, "From User Stories to Models: A Machine Learning Empowered Automation", *9th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), Palaiseau, France*, Vol. 10, pp. 28–40, 2021.

39. Gülle, K. J., N. Ford, P. Ebel, F. Brokhausen and A. Vogelsang, "Topic Modeling on User Stories Using Word Mover's Distance", *IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pp. 52–60, IEEE, 2020.

40. Resketi, M. R., H. Motameni, H. Nematzadeh and E. Akbari, "Automatic Summarising of User Stories in Order To Be Reused in Future Similar Projects", *IET Software*, Vol. 14, No. 6, pp. 711–723, 2020.

41. Elallaoui, M., K. Nafil and R. Touahni, "Automatic Generation of UML Sequence

Diagrams From User Stories in Scrum Process", *10th International Conference on Intelligent Systems: Theories and Applications (SITA), Rabat, Morocco*, pp. 1–6, 2015.

42. Lucassen, G., F. Dalpiaz, J. M. E. M. v. d. Werf and S. Brinkkemper, "Visualizing User Story Requirements at Multiple Granularity Levels via Semantic Relatedness", *Conceptual Modeling: 35th International Conference, ER 2016, Gifu, Japan*, pp. 463–478, 2016.

43. Arora, C., M. Sabetzadeh, L. Briand and F. Zimmer, "Extracting Domain Models From Natural-Language Requirements: Approach and Industrial Evaluation", *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France*, pp. 250–260, 2016.

44. Mesquita, R., A. Jaqueira, M. Lucena, C. Sá Filho and F. M. Alencar, "US2StarTool: Generating i* Models From User Stories", *International i* Workshop, Ottawa, Canada*, pp. 103–108, 2015.

45. Lin, J., H. Yu, Z. Shen and C. Miao, "Using Goal Net To Model User Stories in Agile Software Development", pp. 1–1, 15th IEEE/ACIS International Conference on SNPD, Las Vegas, NV, USA, 2014.

46. Wautelet, Y., S. Heng, M. Kolp, I. Mirbel and S. Poelmans, "Building a Rationale Diagram for Evaluating User Story Sets", *IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), Grenoble, France*, pp. 1–12, 2016.

47. Wautelet, Y., S. Heng, D. Hintea, M. Kolp and S. Poelmans, "Bridging User Story Sets With the Use Case Model", *Advances in Conceptual Modeling: ER 2016 Workshops, AHA, MoBiD, MORE-BI, MReBA, QMMQ, SCME, and WM2SP, Gifu, Japan, Proceedings 35*, pp. 127–138, 2016.

48. Wautelet, Y., S. Heng, S. Kiv and M. Kolp, "User-Story Driven Development of Multi-Agent Systems: A Process Fragment for Agile Methods", *Computer Languages Systems & Structures*, Vol. 50, pp. 159–176, 2017.

49. Wu, C., C. Wang, T. Li and Y. Zhai, "A Node-Merging Based Approach for Generating iStar Models From User Stories", *Software Engineering and Knowledge Engineering*, Vol. 32, pp. 257–262, 2022.

50. Winkler J.P., V. A., "Using Tools To Assist Identification of Non-requirements in Requirements Specifications – A Controlled Experiment", *Requirements Engineering: Foundation for Software Quality: 24th International Working Conference, Utrecht, The Netherlands, Proceedings 24* , pp. 57–71, Springer, 2018.

51. Ko, L. T. . B. M., A.J., "A Practical Guide To Controlled Experiments of Software Engineering Tools With Human Participants", *Empirical Software Engineering*, Vol. 20, pp. 110–141, 2015.

52. Abrahão, S., E. Insfran, F. G. L. Guevara, M. Fernández-Diego, C. Cano Genoves and R. Oliveira, "Assessing The Effectiveness of Goal-Oriented Modeling Languages: A Family of Experiments", *Information and Software Technology*, Vol. 116, pp. 106–171, 2019.

53. Raymond, J. W., E. J. Gardiner and P. Willett, "RASCAL: Calculation of Graph Similarity Using Maximum Common Edge Subgraphs", *The Computer Journal*, Vol. 45, pp. 631–644, 2002.

54. Koutra, D., J. T. Vogelstein and C. Faloutsos, "DELTACON: A Principled Massive-Graph Similarity Function", *Proceedings of the 2013 SIAM International Conference in Data Mining (SDM), Texas, USA*, pp. 162–170, 2013.

55. Koutra, D., J. T. Vogelstein and C. Faloutsos, "Algorithms for Graph Similarity and Subgraph Matching", *Ecological Inference Conference*, Vol. 17, Citeseer, 2011.

56. Zheng, W., L. Zou, X. Lian, D. Wang and D. Zhao, "Efficient Graph Similarity Search Over Large Graph Databases", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, No. 4, pp. 964–978, 2014.