USING SYSTEM MODELS TO RESOLVE CO-REFERENCE IN TRANSLATING NATURAL LANGUAGE REQUIREMENTS INTO CODE

Brit Arnon[1,2], David Harel[1] and Michal Gordon-Kiwkowitz[3]
[1]*Weizmann Institute of Science, Rehovot, Israel*
[2]*Google LLC. Tel-Aviv, Israel*
[3]*Holon Institute of Technology, Holon, Israel*

ABSTRACT

One important quest in software development is to be able to translate natural language requirements into executable code. In this paper we address the issue of co-reference resolution, extending our previous work on a controlled natural language translation to live sequence charts. While domain-specific co-reference resolvers perform adequately, and state-of-the-art deep learning-based generic resolvers are emerging, in software development, especially for critical systems, a higher level of performance is required. We present an approach that incorporates system-model-based anaphora resolution. Using an annotated corpus for software requirements with system models, we show that while our approach reaches high accuracy, it is synergistic to generic resolvers and when combined with them it reaches higher accuracy. Our method shows that incorporating system models into the pipeline of requirements to code is a promising approach.

KEYWORDS

Requirements to Code, Live Sequence Charts, Co-Reference Resolution, Controlled Natural Language

1. INTRODUCTION

In an age when technology is all around and more people are involved in building software, simpler interfaces for development tasks are required [Stefanidi et al., 2018]. A long-term goal of software engineering is to develop tools that translate software requirements directly into executable code, skipping the "programming" phase. There are two approaches to reaching this goal, namely, processing natural language and making software languages more compatible with requirements.

The challenge with software requirements is that they are usually written in natural language (NL) and processing them is challenging. One approach is to use a "controlled natural language" (CNL), which imposes restrictions and rigor on the requirements. In [Bryant & Lee, 2002] the authors present a formal but natural-like language that can translate software requirements into Java code, while [Finucane et al., 2010] features a robot controller that transforms task specifications into direct motor commands to the robot.

However, the programming languages used in these attempts were not developed with requirements in mind. Behavioral programming languages, on the other hand, attempt to be compatible with how people think about software requirements; they are more event-driven and are often visual. One such language is live sequence charts (LSC), which is an extension of message sequence charts [Damm & Harel, 2001].

There have been attempts to create an NL interface for behavioral programming. In [Gordon & Harel, 2009], a novel approach was developed based on interaction with the user, whereas [Tsarfaty et al., 2014] features a machine-learning-based model for translating requirements written in the CNL of [Gordon & Harel, 2009] into LSCs.

One outstanding challenge with these approaches is the resolution of co-reference; that is, deciding which of two or more expressions that appear to refer to the same entity is the correct one. Co-reference resolution is challenging, because it deals with ambiguity. For example, *"when the car approaches the garage, it opens"*: does *it* relate to the car or the garage? While there have been many attempts to overcome this challenge [Shekhar & Kumar, 2018], only very few software requirement translating tools have succeeded

[Yang et al., 2011]. The task of resolving co-references for software requirements has strict demands, more so when the requirements are to be translated into executable code for critical systems. In this paper, we describe a method for co-reference resolution in the CNL interface described in [Gordon & Harel, 2009], translated into executable LSC.

The paper is organized as follows: Section 2 contains preliminaries. We discuss co-reference resolution, followed by a brief description of LSC, a CNL for the language of LSC, and the basics of co-reference resolution in natural language. In Section 3, we present our co-reference resolution method, and in Section 4 we present the results. We then conclude with related work and conclusions in Section 5.

2. PRELIMINARIES

2.1 Co-Reference Resolution

Research on co-reference resolution has focused on various domains separately [Shekhar & Kumar, 2018]. In [Rosiger & Teufel, 2014] the reference resolver of [Bjorkelund & Farkas, 2012] is adapted to the domain of scientific texts, and it is shown that adding domain specific training and knowledge drastically increases the success rate. In [Zheng et al., 2011] a co-reference resolution method that focuses on the clinical domain is reviewed, and [Uzuner et al., 2012] evaluates several such methods. The conclusion is that domain knowledge and semantics-related features are required for satisfactory co-reference resolution. While such features are not prominent in many reference resolution systems, our CNL-specific system presented here uses semantics-related features quite heavily; namely, the system model information.

Despite these advances, general co-reference resolution is still an unsolved problem [Peng et al., 2015]. More recent approaches use deep learning algorithms to address the problem. In [Lee et al., 2017] an end-to-end co-reference resolution model is presented, which uses a recurrent deep neural network. Expansions to this model appear in [Luo & Glass, 2018], where cross-sentence dependencies improve performance. Yet state-of-the-art performance is still below 80%.

Several approaches have focused on processing requirements written in a "free" natural language. In [Nanduri & Rugaber, 1995], a natural language parser is developed, which transforms requirements into an object oriented structure and thus enables some kinds of automatic validation. In [Roth & Klein, 2015] a semantic role labeler is used to convert natural language requirements into structured semantic representations. In [Chhabra et al., 2018], Petri nets and context based reasoning are used to transform natural language requirements into a more formal representation — expressive decision tables.

Our interest here is co-reference resolution in requirement documents. One approach to this appears in [Yang et al., 2011], where a human subject is used to help train a classifier for solving ambiguous co-references in requirement documents. As we shall see, our approach combines system knowledge with requirements parsing to boost the efficiency of co-reference resolution.

2.2 Controlled Natural Language for Live Sequence Charts

Live sequence charts (LSCs) [Damm & Harel, 2001] are a formal visual language for specifying programming scenarios of reactive systems. Each LSC is a diagram that describes a specific scenario of the system: a possible one, a necessary one or a forbidden one. A set of LSCs specifies the behavior of an entire system, and such a set can be directly executed, or compiled into conventional executable code that realizes that behavior [Harel & Marelly, 2003; Harel et al., 2010].

Let us begin with an example. Consider a system that simulates a “smart home”: a central unit that automatically controls the air conditioning and the lighting in the house. We may have certain expectations regarding such a system: for example, we can expect the lights in a room to turn on as people enter and to turn off when the room becomes empty; or we can expect the air conditioner to start and stop according to the current temperature in the room. Such requirements can be expressed easily using LSCs.

Formally, the behavioral requirements expressible in this language involve elements in the structural model of the system: objects, properties, methods and values. We represent these elements in a data structure called **the system model**. The system model contains all the classes and objects of the system. Each class has a set of properties and a set of methods that are associated with this class. Each object is an instance of a certain class, and it acquires the properties and methods of that class. For example, in the case of the smart home, the system consists of several objects: rooms, lights, an air conditioner, blinds, and so on. Each object has properties and methods. For example, a blind has a property state and methods open() and close().

In [Gordon & Harel, 2009] a CNL was developed for expressing requirement scenarios that can be directly translated into LSCs. The language is a simplified version of English; it has a rigid grammar, but it is quite expressive. For example, requirements from the world of the "smart house" can be expressed as follows*: "When Alice enters the room, the light state is set to on"* or *"When Alice enters the room, if the room temperature is greater than 24, the air_conditioner starts"*.

The CNL of [Gordon & Harel, 2009] is structured as a context-free grammar. The terminal symbols in the grammar are of two types, static and dynamic. Static terminals are predefined in the grammar, and are words that describe the flow and the structure of the scenario; e.g., *when*, *then*, *if*. Dynamic terminals are not predefined, and can be written quite freely; they refer to the elements of the system. **Objects** are expressed by noun phrases: *Alice*, *the room*, *the light*, *the air conditioner*. **Properties** are expressed by compound noun phrases, composed of an object and another noun: *the light state*, *the room temperature*. **Constant values** can be expressed by several parts of speech. For example, *on* is a preposition and *24* is a number. Almost any word can be considered a constant value. **Messages**: the CNL distinguishes between two types of messages: method calls and property changes. A **method call** represents a call to an object's method; for example, the phrase *Alice enters the room* expresses a call to the method enter() of the object the room. The method is expressed by a verb. A **property change** is the event of setting the value of a property; for example, the phrase *the light state is set to on* indicates that the property the light state is assigned the value "on". It is actually a call to an object's setter method, but it is usually not expressed as such in the grammar.

We note that in this approach words can be ambiguous. For example, a noun may express either an object or a constant value. Moreover, many English noun forms can be interpreted as verbs (e.g., *light* is both a noun and a verb), so they may express objects, constant values or verbs. These inherent lexical ambiguities cause the language to be highly ambiguous. Yet, combining the CNL translation with LSCs means that we are able to generate executable code (see [Harel & Marelly, 2003]) out of textual requirements in a controlled subset of English.

2.3 Reference Resolution in Natural Language

Reference resolution is the task of determining which entity is referred to by a certain expression in the discourse. For example, in the sentence (taken from our "smart home" domain): *"When Alice enters the room, if **its** temperature is less than 24, the smart home opens all the blinds, otherwise **it** closes **them**."* In this example, *its* refers to *the room*, while *it* refers to the *smart home*, and *them* refers to *all the blinds*.

An expression that refers to an entity is called a **referring expression** or a **mention**, and the entity it refers to is called its **referent**. If two mentions refer to the same entity, we say that they are **co-referent**. In the sentence above the co-references are *the room* and *its*, the *smart home* and *it* and *all the blinds* and *them*.

Note that in the example sentence each model entity is first expressed explicitly, and only then is referred to by a pronoun. This type of reference, where the explicit mention appears before the pronoun, is called **anaphora**. The pronoun is said to be an **anaphoric pronoun**, and the explicit mention is said to be the **antecedent** of the pronoun. The explicit mention may appear after the pronoun, as in the following sentence: *"Whenever **she** enters the room, Alice opens the windows*". This phenomenon is called **cataphora**. It is less common than anaphora, and therefore in this paper we focus on the task of anaphora resolution.

We do not attempt to cover them here, but instead briefly mention some prominent principles in reference resolution that are frequently used, and which are relevant to our work. One is **grammatical agreement**: co-referent expressions should agree in **number** (singular/plural), **gender** (male/female/neuter) and **person** (1st/2nd/3rd). Another is **salience**: the more salient an entity is in the discourse, the more probable it is to be an antecedent of a pronoun. There are several methods for determining how salient a certain entity is in a given sentence [Jurafsky & Martin, 2009].

ISBN: 978-989-8704-15-3 © 2020

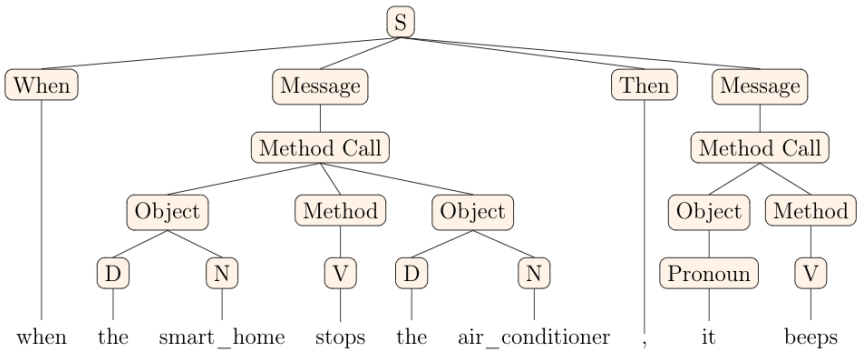

Figure 1. The parse tree (see [Jurafsky & Martin, 2009] for details) for the sentence: *"When the smart_home stops the air_conditioner, it beeps"*. The terminal *it* is generated from the non-terminal *pronoun*

3. ADDING CO-REFERENCE RESOLUTION TO THE CNL

In this section we present our approach to co-reference resolution in sentences written in our CNL. First, since the grammar in [Gordon & Harel, 2009] did not include pronouns, we have extended the grammar to include them. Since pronouns are used to refer to noun phrases, they should appear in the same contexts as noun phrases do, namely as objects and properties. The extension was quite straightforward: we added the non-terminal *pronoun*, which may generate any English pronoun; and the *object* and *property* non-terminals may generate a *pronoun*. As an example, Figure 1 shows the full parse tree of the sentence *"When the smart home stops the air conditioner, it beeps"*.

We note that pronouns in 1st or 2nd person (*I* or *you*) seldom appear in software requirement specifications, and therefore we support only 3rd person pronouns (*she*, *he*, *it*, *they* and their inflections).

Recall from subsection (2.2) that the system model is the collection of the classes and objects associated with methods and properties. As we mentioned in subsection (2.3), a principal technique in reference resolution is enforcing grammatical agreement between the co-referent expressions: they must agree on gender and number. In our approach we extend this, and enforce also "class agreement": co-referent mentions must be associated with the same class. The class can be deduced from methods and properties that are mentioned in the context.

For example, consider the following pair of requirements, written in the CNL: (a) *"When the smart home stops the air conditioner, **its** state changes to off."* (b) *"When the smart home stops the air conditioner, **its** message changes to "stopped the air conditioner"."* In (a), the referent of *it* must have a property named *state*; otherwise the requirement would be nonsensical. Therefore, *it* must refer to *the air conditioner*, which is the only object with such a property. Likewise, in (b), the referent of *it* must have a property named message, and therefore it must be *the smart home*. This constraint can be formulated as follows: **Property-Related Constraint:** If a pronoun *p* is immediately followed by a noun *n*, the referent of *p* is an object, which is an instance of a class that has a property named *n*.

The situation for methods is more complicated. In the CNL, methods can appear in three possible constructs: **(1) Object Method**. Example: *"the air_conditioner beeps"*. The pronoun in the phrase *"it beeps"* must refer to an object that has a method named *beep*. **(2) Object Method Property**. Example: *"the smart_home records the room temperature"*. The pronoun in the phrase *"it records the room temperature"* must refer to an object that has a method named *record*. **(3) Object Method Object**. Example: *"the smart_home stops the air_conditioner"*. Here a pronoun can replace either of the two objects. However, according to the CNL grammar, the "agent" of the method in this structure is always the second object. Thus, the *stop* method mentioned is a method of the *air_conditioner*.

This analysis can be formulated as the following constraint: **Method-Related Constraint**: For a pronoun *p*, if a verb *v* is associated with *p*, the referent of *p* is an object that is an instance of a class that has a method named *v*.

In the **system model inference** technique, we extract system-model-related attributes of the entities in the sentence, and then use them to impose constraints on potential co-referent entities. The attributes are methods and properties that are associated with the objects. We note that these considerations do not always suffice to determine the antecedent of a pronoun.

Our goal was to adapt the system of (Gordon & Harel, 2009), which translates requirements written in the CNL into LSCs, to be able to deal with sentences that include referring pronouns. The original system consists of two components: a parser and a semantic processor; We added an anaphora resolution component, which uses the system model inference technique. We now describe the operation of the anaphora resolver and the modifications we made in the other components.

The **parser** receives a single sentence in the CNL as input, and outputs the most probable parse tree for this sentence. Our parser also processes pronouns. The **semantic processor** interacts with the anaphora resolver. Whenever it encounters a pronoun, it requests the anaphora resolver to resolve it.

The **anaphora resolver** is our major contribution. Given a parsed sentence and a pronoun therein, the anaphora resolver outputs an appropriate antecedent. In order to do so, it maintains a set of all the explicit entities that have been extracted by the semantic processor. For a given pronoun, the anaphora resolver constructs a set of extracted entities that are compatible antecedents according to the system model constraints, and outputs one of them arbitrarily. (The optimal case is that the resulting set has only one element, which is the correct antecedent.) The algorithm is "system-model-based" (**SMB**) and is shown in Algorithm 1. Lines 3-4 enforce the method-related constraint on the object entities, and lines 5-6 enforce the property-related constraint on them. Notice that at most one of them is applied (since, grammatically, a pronoun cannot be both associated with a verb and immediately followed by a noun). Lines 8-9 handle property entities: since there are no constraints for properties, the algorithm simply outputs the first property entity.

This simple algorithm uses almost no considerations other than the system model constraints. As our analysis will show, its performance is encouraging. Furthermore, it adds a feedback mechanism: An additional benefit since for certain types of incorrectly parsed sentences it can detect the error by using the system model information. Having not found any appropriate antecedent for a pronoun in the sentence is an indication that the sentence's parse tree is incorrect. In such cases, the anaphora resolver notifies the parser, which then selects another parse tree, and the semantic processing is restarted. This feedback mechanism allows for a higher accuracy of the entire system.

Algorithm 1: System-model-based (SMB) anaphora resolution algorithm

Input: a parsed sentence *s*, and an anaphoric pronoun *p* from *s*.

Denote by $O = \{o_1, \ldots, o_k\}$ the set of extracted objects, and denote by $R = \{r_1, \ldots, r_k\}$ the set of extracted properties. Assume that the entities are sorted according to their order of appearance in the sentence.

1: **if** *p* is parsed as an object in *s* **then**
2: construct a set *O'* which is a copy of *O*.
3: **if** a verb *v* is associated with *p* **then**
4: remove from *O'* any object that does not have a method named *v*.
5: **else if** *p* is immediately followed by a noun *n* **then**
6: remove from *O'* any object that does not have a property named *n*.
7: **return** the first entity in *O'*.
8: **else if** *p* is parsed as a property in *s* **then**
9: **return** the first entity in *R*.

4. RESULTS

As far as we could tell, there is no corpus of requirement specification texts with reference annotations. Therefore, we built and annotated our own dataset. The collection consists of several sets of LSC programs and the CNL sentences that correspond to them. We replaced recurring mentions in these sentences with pronouns. In some cases we joined closely-related sentences into a paragraph, or replaced a word with a synonym in order to make the sentence clearer. However, we made sure not to alter the grammatical structure of the sentences, so that the resulting dataset is an accurate reflection of the original collection.

The dataset consists of 68 paragraphs, which include 107 pronouns, and we mapped each pronoun to its antecedent. We considered several variants of anaphora resolution algorithms and compared their performance on our dataset. Each system received a binary score on each pronoun, indicating whether the system outputs the correct antecedent for it.

As a baseline, we used a known reference resolution system, developed by [Lee et al., 2011], which is a part of the Stanford CoreNLP toolkit [Manning et al., 2014]. It has two variants: a statistical one (machine-learning-based) and a deterministic one (rule-based). Note that the system output is not a pronoun-antecedent mapping, but a set of clusters of co-referent mentions, so it had to be adapted for the needs of our evaluation. We added an antecedent selection layer: for each pronoun, the layer finds the cluster that contains it and selects the cluster's first mention (by order of appearance in the text) to be the antecedent. We denote the statistical and deterministic variants by **SST**[1] and **SDE**[2], respectively.

We considered several variants of our system-model-based algorithm. The first is the SMB[3] algorithm described in the previous section**.**.

Next, recall that this basic algorithm filters out incompatible antecedent candidates, and then selects one of the remaining candidates arbitrarily. The selection phase is completely naive, since the system model information can be used to reject candidates but not to score the remaining ones. In order to make the selection phase more sophisticated, we extended the algorithm by "consulting" the Stanford CoreNLP systems, in the following way. In the selection phase the extended algorithm examines two sets: the candidate set constructed in the filtering phase, and the set of mentions that the Stanford system variant considers to be co-referent with the given pronoun. If possible, the mechanism outputs a mention that is included in both sets. This method refines the candidate set, and thus we expected it to improve the accuracy of the selection phase. We used two variants, one that consults the Stanford deterministic system, denoted **SMBcSDE**[4], and one that consults the statistical system, denoted **SMBcSST**[5]. Note that SMB uses only system model knowledge, while the Stanford systems use various syntactic and structural features. This makes the two types of systems complementary, and therefore their integration is expected to be beneficial.

Table 1 compares the performance of the three anaphora resolution algorithms. The SMB variants outperform SDE significantly, and the "consulting" variants outperform the "non-consulting" one. One could claim that the main reason for this is that SMB has been specifically developed to handle the grammar in which the requirements are written. However, we claim that incorporating system model information directly is especially beneficial. To support this claim, we examined the impact of incorporating the system model inference into SDE. (For more details on SDE, see [Lee et al., 2011].)We enriched SDE's pronominal co-reference model so that it enforces agreement regarding two additional attributes: (1) the type of the mention: object, property or value; and (2) the set of possible classes of which the mention can be an instance, if it is an object. Two mentions are considered to agree on these attributes if they share their type and if their class sets intersect. This agreement enforcement was implemented using the same scheme by which the grammatical agreement enforcement is implemented in the original system.

Another necessary modification was in the mention extraction process. As described in [Lee et al., 2011], at the beginning of the process the model scans the text and extracts all noun phrases that it considers to be candidate mentions. The extraction is fairly strict: it skips noun phrases that are less likely to refer to world entities, such as phrases that are preceded by quantifiers, like *all* and *every*. These considerations are very suitable for general texts, but less suitable for requirement specifications. We modified the mention extraction process, making it more relaxed for noun phrases that are typically used in requirement specifications.

We incorporated the system model information into SDE, and not into SST, since the way to adapt SST would be by training it on an appropriate dataset, which should be sufficiently large. We hope that such a dataset will be constructed in the future.

[1] SST - Stanford Statistical Variant.
[2] SDE – Stanford Deterministic Variant.
[3] SMB – System Model Based Variant.
[4] SMBcSDE – System Model Based consulting the Stanford Deterministic Results
[5] SMBcSST – System Model Based consulting the Stanford Statistical Results.

Table 1 also shows the performance of three variants of SDE: the original one, one that has relaxed mention extraction, denoted SDE-R[6], and one that has relaxed mention extraction and uses system model inference, denoted SDE-R-SM[7]. Incidentally, the accuracy of the variant that uses system model inference with the original strict mention extraction was similar to the original SDE version, since the correct antecedents were not extracted as mentions, and hence were not considered as antecedents. It is therefore omitted from the table. The relaxed mention extraction allows for more mentions to be extracted and assigned as antecedents, thus improving the accuracy. For example, SDE produces an incorrect result when applied to the paragraph *"When the traffic light status is set to "green", all cars with speed 0 resume. When **they** resume, **their** speed is set to the controller speed"*. In contrast, SDE-R gives the correct result on it, since it extracts *all cars with speed 0* as a mention and considers it as an antecedent to *they* and *their*.

Table 1. The accuracy of the anaphora resolution algorithms on the left and of the three variants of the Stanford deterministic reference (SDE) resolution combinations with the system model on the right. The table shows the percentage of correct outputs out of 107

System Model Based Algorithm Variants	% correct
SMB[3]	81.3
SMBcSDE[4]	82.24
SMBcSST[5]	87.85

SDE Variants	% correct
SDE[2]	44.85
SDE-R[6]	54.2
SDE-R-SM[7]	68.22

The system model inference technique allows for selecting more accurate antecedents. For example, SDE-R is incorrect on the sentence *"When the smart home stops the air conditioner, **its** state is set to off."*, while SDE-R-SM is correct on it: it does not allow *the smart home* to be co-referent with *it*, since the smart home does not have a property state, and therefore *the air conditioner* is correctly recognized as co-referent with *it* and as its antecedent. We thus see that adapting SDE to the domain and incorporating system model information improves its performance.

5. CONCLUSION

We have presented a novel integration of a controlled natural language parser with a system model-based technique for anaphora resolution. While several methods have been proposed for co-reference resolution [Rosiger & Teufel, 2014; Zheng et al., 2011], they are usually domain-specific, and very rarely address software requirements [Roth & Klein, 2015]. However, there are several generic co-reference resolution methods [Lee et al., 2011, 2017], and our approach is synergistic with these, as we have demonstrated with the out-of-the-box Stanford Core toolkit resolver [Lee et al., 2011].

We have shown that augmenting a generic resolver with our system model based resolver drastically improves performance. Hence, we suggest that state-of-the-art algorithms can also benefit greatly from our approach. One limitation is that the current work was tested on a CNL that may not be appropriate for all domains.

Another direction for future work, stems from work like that of [Arora et al., 2016], where a method is proposed for extracting the model of a system from its requirements. We can use such a method in the future to extract the system model automatically, instead of constructing it manually as we did. And in a totally different direction, in [Mitkov et al., 2007] it is shown that deploying an anaphora resolution module can improve the performance of other NLP applications. This encourages us to believe that our anaphora resolution mechanism is the first step in extending our CNL-to-LSC system with more advanced features.

In summary, co-reference resolution in requirements engineering must be precise, since the end goal is an executable system. The presented method, built upon a controlled natural language and a co-reference resolution algorithm based on the system model, appears to be a promising new approach.

[6] SDE-R – Stanford Deterministic Variant with Relaxed mention extraction
[7] SDE-R-SM – Stanford Deterministic Variant with Relaxed mention extraction and uses System Model inference

REFERENCES

Arora, C., Sabetzadeh, M., Briand, L., & Zimmer, F., 2016. Extracting domain models from natural-language requirements: approach and industrial evaluation. *Proc. of the ACM/IEEE 19th Int. Conf. on Model Driven Engineering Languages and Systems*. Saint-malo, France, pp. 250-260.

Bjorkelund, A. & Farkas, R., 2012. Data-driven Multilingual Coreference Resolution using Resolver Stacking. Joint Conf. on EMNLP and CoNLL - Shared Task. Jeju Island, Korea, pp. 49–55.

Bryant, B. R. & Lee, B.-S., 2002. Two-level grammar as an object-oriented requirements specification language. *Proc. of the 35th Annual Hawaii Int. Conf. on System Sciences (HICSS)*. Hawaii, pp 280.

Chhabra, A., Sangroya, A., & Anantaram, C., 2018. Formalizing and Verifying Natural Language System Requirements using Petri Nets and Context based Reasoning. *MRC@ IJCAI*. Stockholm, Sweden, pp. 64–71.

Damm, W. and Harel, D., 2001, LSCs: Breathing Life into Message Sequence Charts. *Formal Methods in System Design*, Vol. 19, No. 1, pp 45-80.

Finucane, C., Jing, G., and Kress-Gazit, H., 2010, LTLMoP: Experimenting with language, temporal logic and robot control. *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. pp 1988-1993.

Gordon, M. and Harel, D., 2009, Generating Executable Scenarios from Natural Language. *Proc. of the 10th Int. Conf. on Computational Linguistics and Intelligent Text Processing (CICLing)*. Mexico, pp 456-567.

Harel, D., et al., 2010. PlayGo: Towards a Comprehensive Tool for Scenario Based Programming. *Proc. of the 25th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE)*. Antwerp, Belgium, pp. 359-360.

Harel, D. & Marelly, R., 2003. *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer-Verlag.

Jurafsky, D. & Martin, J. H., 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall.

Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M., & Jurafsky, D., 2011. Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL – 2011 Shared Task. *Proc. of the Conf. on Natural Language Learning (CoNLL)*. Portland, Oregon, USA, pp. 28-34.

Lee, K., He, L., Lewis, M., & Zettlemoyer, L. S., 2017. End-to-end Neural Coreference Resolution. *Proc. of the 2017 Conf. on Empirical Methods in Natural Language Processing, EMNLP*. Copenhagen, Denmark, pp. 188-197.

Luo, H. & Glass, J., 2018. Learning Word Representations with Cross-Sentence Dependency for End-to-End Co-reference Resolution. *Proc. of the 2018 Conf. on Empirical Methods in NLP*. Brussels, Belgium, pp. 4829–4833.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D., 2014. The Stanford CoreNLP Natural Language Processing Toolkit. *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*. Baltimore, Maryland, pp. 55-60.

Mitkov, R., Evans, R., Orasan, C., Pekar, V., & others, 2007. Anaphora resolution: To what extent does it help NLP applications? *Discourse Anaphora and Anaphor Resolution Colloquium*. Lagos, Portugal, pp. 179–190.

Nanduri, S. & Rugaber, S., 1995. Requirements Validation via Automated Natural Language Parsing. *Journal of Management Information Systems*. Vol. 12, No. 3, pp. 9–19.

Peng, H., Khashabi, D., & Roth, D., 2015. Solving Hard Coreference Problems. *Proc. of the 2015 Conf. of the North American Chapter of the ACL: Human Language Technologies*. Denver, Colorado, pp. 809-819.

Rosiger, I. & Teufel, S., 2014. Resolving Coreferent and Associative Noun Phrases in Scientific Text. *Proc. of the Student Research Workshop at the 14th Conf. of the European Chapter of the Association for Computational Linguistics (EACL)*. Gothenburg, Sweden, pp. 45–55.

Roth, M. & Klein, E., 2015. Parsing Software Requirements with an Ontology-based Semantic Role Labeler. *Proc. Of the 1st Workshop on Language and Ontologies (ACL)*, London, UK.

Shekhar, S. & Kumar, U., 2018. Review on the Techniques of Anaphora Resolutions. *Int. Journal of Latest Trends in Engineering and Technology*, Vol. 8, No. 2, pp. 37–40.

Stefanidi, E., et al., 2018. Programming Intelligent Environments in Natural Language: An Extensible Interactive Approach. *Proc. of the 11th Pervasive Technologies Related to Assistive Environments Conf*. Greece, pp. 50–57.

Tsarfaty, R., et al., 2014. Semantic Parsing Using Content and Context: A Case Study from Requirements Elicitation. *Proc. Int. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pp. 1296–1307.

Uzuner, O., Bodnari, A., Shen, S., Forbush, T., Pestian, J., & South, B. R., 2012. Evaluating the state of the art in coreference resolution for electronic medical records. *J. of the American Medical Informatics Ass*. Vol. 19, No. 5.

Yang, H., de Roeck, A., Gervasi, V., Willis, A., & Nuseibeh, B., 2011. Analysing anaphoric ambiguity in natural language requirements. Requirements Engineering. Vol. 16, No. 3, pp. 163.

Zheng, J., Chapman, W. W., Crowley, R. S., & Savova, G. K., 2011. Coreference resolution: A review of general methodologies and applications in the clinical domain. J. of Biomedical Informatics. Vol. 44, No. 6, pp. 1113-1122.