Advanced Engineering Informatics 28 (2014) 28–36

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Freedom through constraints: User-oriented architectural design

R.A. Niemeijer, B. de Vries [*], J. Beetz

Eindhoven University of Technology, P.O. Box 513, 5600 MB, The Netherlands

ARTICLE INFO

Article history:
Received 1 May 2012
Received in revised form 22 July 2013
Accepted 12 November 2013
Available online 5 December 2013

Keywords:
Natural Language Processing
Design constraints
Automated interpretation

ABSTRACT

In this article we report on validated research for the construction of design constraints by automated interpretation of natural language input. We show how our approach of dynamic reconfigurations of parsed syntax trees using a number of production rules is used to formalize and transform natural language constructs into computable constraints that are applied to concrete building information models. The calibration and validation of the proposed algorithms and production rules is based on two test data sets: The verbatim text of Dutch building code regulations for dormer extensions on existing roofs and constraints formulated ad hoc by design students based upon a series of example designs. We show how a prototypical implementation of our approach can be used to interpret 44% of the test data without human interference and how the remaining sentences can be interpreted with minimal additional effort or further development.

1. Introduction

Currently, customers buying a new house are typically limited to two options: either an architect's design is bought without modifications, or a discrete set of limited design alternatives is offered, such as two different kitchen types or the optional addition of a dormer. This customization, however, is very limited, as all design alternatives offered by the architect have to be completely designed up front. Consequently, customers do not get the exact house they want. In many cases this means that house owners immediately start remodelling after the house has been built to get the house they actually wanted. This is a very inefficient state of affairs, leading to unnecessary increases in cost and waste. It would be preferable for buyers to be able to make more extensive changes to the design of the building in the design phase already, so that they can get the house they want, eliminating the need for an additional remodelling step.

In many different industries, the ability to customize a product has become commonplace, with examples ranging from fast food to clothing to the car industry. In the building industry, however, adoption of this practice has lagged behind. At least two reasons for this lack of customization can be identified: (a) The tradition of the architect being the sole designer of the product and an implied artistic autonomy. (b) The large amount of regulations that apply to buildings. Over time, mass customization and participatory design have been applied to the building and construction sector [36,35]. In most cases, though, the amount of flexibility is limited, since the two traditional ways of offering mass customized housing—entirely customized design or choices from predesigned alternatives—result in a trade-off between the freedom of choice among design alternatives and the amount of time required to design them. When creating a design for a consumer product, many rules must be obeyed [18] which in the case of buildings stem from building codes, regulations and design requirements. In this paper, rules are referred to as constraints [26,38,15] composing a Constraint Satisfaction Problem (CSP), that [37] defined as “a problem composed of a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restricts the values the variables can simultaneously take.”

In current practice, most of these constraints are checked manually which results in labor-intensive and error-prone processes. Hence, the automation of formulating, processing and checking of constraints has been of great interest for researchers and practitioners from early years of computational support onwards. Initial contributions by Fenves [13], Fenves and Garret [14] have sparked a large body of research in this area. These include the developments of AI-based Expert Systems [34,20] as well as Knowledge Based Systems based on Frames [11,10] and Predicate, First Order and Description Logic [33,17]. With the advent of object-oriented Building Information Models (BIM) and particularly the Industry Foundation Classes (IFC), such systems have flourished considerably [19,24,40]. Recently, the incorporation of methods and tools from the Semantic Web initiative led to further advancements in this research field [3,39,32,41]. A concise overview of these developments as well as commercial implementations in the building industry can be found in [12].

* Corresponding author. Tel.: +31 402472388.
E-mail address: B.d.Vries@tue.nl (B. de Vries).

1474-0346/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.aei.2013.11.003

Most of these developments, however, are based on complex logic languages or conventional high-level programming, only allowing users with considerable ICT knowledge to specify, formalize and encode constraints. Secondly, most requirements engineering systems demand prior domain knowledge embedded as corpora in databases. In order to enable ad hoc, per-project formulations of computable constraints by design and engineering end-users, alternative methods are required to address these needs. The goal of this research is to find a method of constraint formulation and entry that is usable by end-user practitioners such as engineers and architects and that generates unambiguous computer interpretable constraints. The research builds upon existing Natural Language Processing (NLP) techniques and provides new technological insights and contributions in the context of architectural design constraints.

The outline of this paper is as follows. First we will discuss NLP in requirements engineering, NLP in building constraint entry and our approach: parsing without a corpus. Following that, we present our prototype system "ContraintSoup". The system's algorithms and components that constitute the prototype are explained in detail. The interface of the prototype is discussed only briefly. We then report the results of a series of experiments measuring the ability of automated interpretation of architectural constraints provided in the form of natural language as the input to the system. Finally, conclusions are drawn on the advantages of the proposed method and how the system can be developed further for improved performance.

2. Constraint and rule formalization

2.1. Natural Language Processing in requirements engineering

Natural Language Programming (NLP) originates from the desire in software development to use natural language instead of computer scientist specific programming language such as Java and C#. The advantages are obvious, namely a direct interaction between the domain expert and the computer that executes the expresses procedures. Despite the fact that many 'high level programming languages' have been developed automatic programming is still unresolved. [2] summarized four components of automatic programming: a means of acquiring a high-level specification (requirements), a mechanism for requirements validation, a means of translating the high-level specification into a low-level specification, and an automatic compiler for compilation of the low-level specification. Since then, so-called Very High Level Languages (VHLL) were developed, such as BIDL [27] that were in fact pseudo-natural languages with unambiguous syntax and semantics. At that time with the advent of Object Oriented Programming (OOP), many researchers (e.g. [31]) NLP in requirements engineering is seen as a part of the OO model generation process. This process can be viewed as a sequence of processing steps that starts from a raw text and proceeds to computer interpretable code. Berzins et al. [4] provide a model for NLP that recognizes four steps: (1) tokenization, (2) synthetic parsing, (3) semantic processing, and (4) pragmatics.

The first part of this process is called Part-of-Speech Tagging [5], and is performed by using a large sample of tagged text, which is referred to as a corpus [22,6,28,8]. In tagged text, every word has been assigned its proper part of speech. For every word in the input text, the algorithm searches the corpus to find the correct part of speech for that word.

The second step of NLP analyses larger chunks of a sentence than individual words. The POS information from the previous level is used but in combination with preceding and succeeding words. Using the semantic meaning of some of these words, hierarchical trees are constructed. Parsing methods apply often statistics and rely upon a corpus of training data (words) that are labelled with a specific meaning necessary for tree construction.

The third step is semantic processing. In this step ambiguity, which is common in natural language, is addressed. An example often used to illustrate this is the following sentence:

"Time flies like an arrow, but fruit flies like a banana."

The first occurrence of "flies" is a verb, but in the correct interpretation the second occurrence is a noun. This is an example of a case where one word has more than one possible part of speech, as hinted on in the paragraph on corpora. The deterministic approach stops working here, since the interpretation involving flying fruit would be grammatically correct, but incorrect from the perspective of common sense. The common solution for this problem is to use a statistical approach, where each production rule of the grammar (a rule that governs how parts of the sentence are combined, e.g. verb + adverb = verb phrase) is also attributed with a relative frequency with which it occurs. These types of grammars are referred to as Probabilistic Context-Free Grammars (PCFG) or Stochastic Context-Free Grammars (SCFG) [7,8]. Using these frequencies, the parser can check the neighboring words to determine the likelihood of a given interpretation scenario.

Finally, in the fourth step (pragmatics) is concerned with the more complex linguistics issues, such as resolving what a pronoun or noun refers to.

In [4], four challenges for using NLP in requirements engineering are listed:

(1) Ambiguity of word meaning and scope: words can have multiple meanings, and phrases and adjectives can refer to multiple words.
(2) Computational complexity: the possible need to check an exponential number of parse trees.
(3) Tacit knowledge and anaphora resolutions: the difficulty in resolving reference words such as 'they' when there are multiple possible targets without knowledge of the properties and behaviors of those targets.
(4) Non-linguistic context: information about the stakeholder, time of day, recent events, etc.

When it comes to word meaning, ambiguities can be partially resolved due to the fact that the system operates in a domain-specific context. The word column, for instance, could refer to a vertical list of figures in a spread sheet. In the context of building constraints, however, it is considerably more likely to refer to a pillar. Similarly, the variation in non-linguistic context is limited since all text processed with the system will be a constraint. The main contributions of the proposed algorithm lie in points 2 and 3. By using unit information and typical sentence structures of architectural constraints, it is possible to resolve references and other gaps in the parse tree without taking exponential time.

2.2. Natural Language Processing for building constraint entry

Currently, the majority of constraints in the building industry are specified using a natural language, such as Dutch or English. Examples of these include building codes and functional requirements in the client's brief. The inherent complexity and flexibility of natural languages, however, makes automated interpretation exceedingly difficult, as it requires not just an understanding of grammar, but also knowledge of a domain and a sense of context. Although NLP has been applied in building and construction in a large variety of areas, among which information extraction from documents is the most prominent (see [25] for an overview), only a limited amount of research has been dedicated to automate the