DOCUMENTATION PARSER TO EXTRACT SOFTWARE TEST CONDITIONS

Patricia Lutsky
Brandeis University
Digital Equipment Corporation
111 Locke Drive LMO2-1/L11
Marlboro, MA 01752

OVERVIEW

This project concerns building a document parser that can be used as a software engineering tool. A software tester's task frequently involves comparing the behavior of a running system with a document describing the behavior of the system. If a problem is found, it may indicate an update is required to the document, the software system, or both. A tool to generate tests automatically based on documents would be very useful to software engineers, but it requires a document parser which can identify and extract testable conditions in the text.

This tool would also be useful in reverse engineering, or taking existing artifacts of a software system and using them to write the specification of the system. Most reverse engineering tools work only on source code. However, many systems are described by documents that contain valuable information for reverse engineering. Building a document parser would allow this information to be harvested as well.

Documents describing a large software project (i.e. user manuals, database dictionaries) are often semi-formatted text in that they have fixed-format sections and free text sections. The benefits of parsing the fixed-format portions have been seen in the CARPER project (Schlimmer, 1991), where information found in the fixed-format sections of the documents describing the system under test is used to initialize a test system automatically. The current project looks at the free text descriptions to see what useful information can be extracted from them.

PARSING A DATABASE DICTIONARY

The current focus of this project is on extracting database related testcases from the database dictionary of the XCON/XSEL configuration system (XCS) (Barker & O'Connor, 1989). The CARPER project is aimed at building a self-maintaining database checker for the XCS database. As part of its processing, it extracts basic information contained in the fixed-format sections of the database dictionary.

This project looks at what additional testing information can be retrieved from the database dictionary. In particular, each attribute description contains a "sanity checks" section which includes information relevant for testing the attribute, such as the format and allowable values of the attribute, or information about attributes which must or must not be used together. If this information is extracted using a text parser, either it will verify the accuracy of CARPER's checks, or it will augment them.

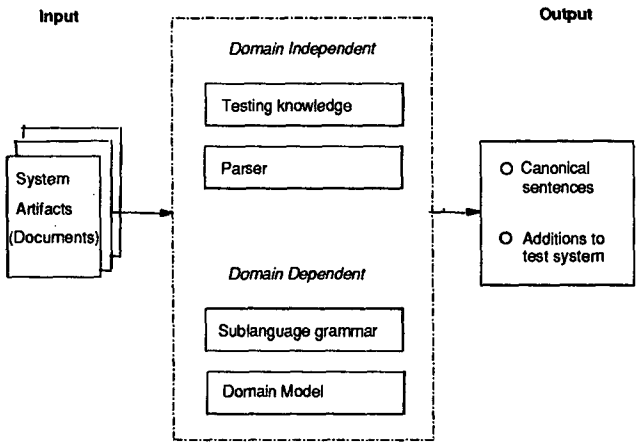
The database checks generated from a document parser will reflect changes made to the database dictionary automatically. This will be particularly useful when new attributes are added and when changes are made to attribute descriptions.

(Lutsky, 1989) investigated the parsing of manuals for system routines to extract the maximum allowed length of the character string parameters. Database dictionary parsing represents a new software domain as well as a more complex type of testable information.

SYSTEM ARCHITECTURE

The overall structure of the system is given in Figure 1. The input to the parser is a set of system documents and the output is testcase information. The parser has two main domain-independent components, one a testing knowledge module and one a general purpose parser. It also has two domain-specific components: a domain model and a sublanguage grammar of expressions for representing testable information in the domain.

Figure 1
Document Parser System



For this to be a successful architecture, the domain-independent part must be robust enough to work for multiple domains. A person working in a new domain should be given the framework and have only to fill in the appropriate domain model and sublanguage grammar.

The grammar developed does not need to parse the attribute descriptions of the input text exhaustively. Instead, it extracts the specific concepts which can be used to test the database. It looks at the appropriate sections of the document on a sentence-by-sentence basis. If it is able to parse a sentence and derive a semantic interpretation for it, it returns the corresponding semantic expression. If not, it simply ignores it and moves on to the next sentence. This type of partial parsing is well suited to this job because any information parsed and extracted will usefully augment the test system. Missed testcases will not adversely impact the test system.

COMBINATION CONDITIONS

In order to evaluate the effectiveness of the document parser, a particular type of testable condition for database tests was chosen: legal combinations of attributes and classes. These conditions include two or more attributes that must or must not be used together, or an attribute that must or must not be used for a class.

The following are example sentences from the XCS database dictionary which concern these test conditions.

1. If BUS-DATA is defined, then BUS must also be defined.

2. Must be used if values exist for START-ADDRESS or ADDRESS-PRIORITY attributes.

3. This attribute is appropriate only for class SYNC-COMM.

4. The attribute ABSOLUTE-MAX-PER-BUS must also be defined.

Canonical forms for the sentences were developed and are listed in Figure 2. Examples of sentences and their canonical forms are given in Figure 3. The canonical form can be used to generate a logical formula or a representation appropriate for input to the test system.

Figure 2
Canonical sentences

```
ATTRIBUTE must [not] be defined if
   ATTRIBUTE is [not] defined.
ATTRIBUTE must [not] be defined for
   CLASS.
ATTRIBUTE can only be defined for
   CLASS.
```

Figure 3
Canonical forms of example sentences

```
Sentence:
   If BUS-DATA is defined then BUS must
   also be defined.
Canonical form:
   BUS must be defined if BUS-DATA is
   defined.

Sentence:
   This attribute is appropriate only
   for class SYNC-COMM.
Canonical form:
   BAUD-RATE can only be defined for
   class SYNC-COMM.
```

THE GRAMMAR

Since we are only interested in retrieving specific types of information from the documentation, the sublanguage grammar only has to

cover the specific ways of expressing that information which are found in the documents. As can be seen in the list of example sentences, the information is expressed either in the form of modal, conditional, or generic sentences.

In the XCS database dictionary, sentences describing legal combinations of attributes and classes use only certain syntactic constructs, all expressible within context-free grammar. The grammar is able to parse these specific types of sentence structure.

These sentences also use only a restricted set of semantic concepts, and the grammar specifically covers only these, which include negation, value phrases ("a value of,") and verbs of definition or usage ("is defined," "is used"). They also use the concepts of attribute and class as found in the domain model. Two specific lexical concepts which were relevant were those for "only," which implies that other things are excluded from the relation, and "also," which presupposes that something is added to an already established relation. The semantic processing module uses the testing knowledge, the sublanguage semantic constructs, and the domain model to derive the appropriate canonical form for a sentence.

The database dictionary is written in an informal style and contains many incomplete sentences. The partially structured nature of the text assists in anaphora resolution and ellipses expansion for these sentences. For example, "Only relevant for software" in a sanity check for the BACKWARD-COMPATIBLE attribute is equivalent to the sentence "The BACKWARD-COMPATIBLE attribute is only relevant for software." The parsing system keeps track of the name of the attribute being described and it uses it to fill in missing sentence components.

EXPERIMENTAL RESULTS

Experiments were done to investigate the utility of the document parser. A portion of the database dictionary was analyzed to determine the ways the target concepts are expressed in that portion of the document. Then a grammar was constructed to cover these initial sentences. The grammar was run on the entire document to evaluate its recall and precision in identifying additional relevant sentences. The outcome of the run on the entire document was used to augment the grammar, which can then be run on successive versions of the document over time to determine its value.

Preliminary experiments using the grammar to extract information about the allowable XCS attribute and class combinations showed that the system works with good recall (six of twenty-six testcases were missed) and precision (only two incorrect testcases were returned). The grammar was augmented to cover the additional cases and not return the incorrect ones. Subsequent versions of the database dictionary will provide additional data on its effectiveness.

SUMMARY

A document parser can be an effective software engineering tool for reverse engineering and populating test systems. Questions remain about the potential depth and robustness of the system for more complex types of testable conditions, for additional document types, and for additional domains. Experiments in these areas will investigate deeper representational structures for modal, conditional, and generic sentences, appropriate domain modeling techniques, and representations for general testing knowledge.

ACKNOWLEDGMENTS

I would like to thank James Pustejovsky for his helpful comments on earlier drafts of this paper.

REFERENCES

Barker, Virginia, & O'Connor, Dennis (1989). Expert systems for configuration at DIGITAL: XCON and beyond. Communications of the ACM, 32, 298-318.

Lutsky, Patricia (1989). Analysis of a sublanguage grammar for parsing software documentation. Unpublished master's thesis, Harvard University Extension.

Schlimmer, Jeffrey (1991) Learning meta knowledge for database checking. Proceedings of AAAI 91, 335-340.