Automatic Generation of a Software Requirements Specification (SRS) Document

Marinos G. Georgiades
Department of Computer Science
University of Cyprus
Nicosia, Cyprus
e-mail: colognet@ucy.ac.cy

Andreas S. Andreou
Dept. of Electr. Engineering & Information Technology
Cyprus University of Technology
Limassol, Cyprus
e-mail: andreas.andreou@cut.ac.cy

***Abstract*—The lack of a software tool to automate the creation of a well-defined, Natural Language Software Requirements Specification (SRS) document is an obstacle to the process of efficient Requirements Engineering (RE). This paper provides an overview of a methodology and a novel software tool that attempt to formalize and automate the RE process, and it expands on the use of the SRS Documentation component of the tool that generates automatically a well-structured Natural Language SRS document.**

Keywords- automated SRS; automated RE; Natural Language RE

I. INTRODUCTION

A Software Requirements Specification (SRS) is a detailed description of the functional and non-functional requirements of the system to be developed. It is also the analyst's understanding (in writing) of the client's system requirements prior to design [1]. It also serves as a mutual agreement between the client and the analyst (organisation) about what the customer wants and what the analyst will advance for development. The SRS has to be written precisely and explicitly in natural language, following a well-structured organisation; this is done with the use of the SRS document.

Software requirements specification is a difficult and complex task that requires considerable effort from the requirements engineer [2]. To create the SRS document, the analyst usually uses requirements specification templates that provide a general structure of the functional and non-functional requirements of the new Information System (IS). However these templates do not provide specific direction that is linked to the requirements discovery stage. Instead, the analyst needs to search through documents, or to make interviews with open questions, and the result is a long, disorganized text, with redundancies and ambiguities, which the analyst needs to organize and fit to the requirements specification template. Apart from the lack of specificity within the SRS documents, there is also lack of automatic generation of the SRS document. There are many Computer Aided Software Engineering (CASE) tools that help with the development of software, but they rarely provide support for the natural language (NL) descriptions of requirements. The primary focus of existing tools lies on diagrams, charts and pictures. Less emphasis is given on the textual specification of requirements [3]. However, the need for NL-oriented CASE tools is clear, since NL gives expressiveness to the formalization and writing of requirements and makes them easily understood by the users, analysts and programmers.

For the automatic generation of a specific SRS document, we introduce NALASS (Natural Language Syntax and Semantics), a supporting tool for the formalization and automation of major parts of Requirements Engineering. NALASS consists of a set of components that formalize major parts of Requirements Discovery (RD), Requirements Analysis (RA) and Requirements Specification (RS). The current paper provides an overview of these components/features, on the one hand, and describes in detail, on the other hand, the SRS document generation component aimed at automating software documentation. This component generates the SRS document automatically based on the input provided by the other components. For example the RD, RA and RS components would extract specification patterns in the form of formalized sentential requirements that should be fed into the SRS documentation component which would guide and support the automated generation of the SRS document. The structure of our SRS document covers the essential parts of the IEEE SRS template [4].

The remaining of the paper is organized as follows: Section 2 outlines related work and describes how our tool differs from others. Section 3 presents a general overview of the methodology and the tool, whereas section 4 expands on the documentation component and provides examples of using NALASS for requirements specification. Finally, Section 5 provides conclusions and directions of future work.

II. RELATED WORK

Generally there is lack of CASE tools for automating the major parts of the RE process including discovery, analysis and specification of requirements. In particular, there is also lack of CASE tools that can produce textual descriptions of requirements and embed them automatically in a well-structured SRS document template. Tools such as Rational Rose [5] and MagicDraw [6] provide significant capabilities for drawing diagrams and even generate code from software models but not adequate facilities for the above – hence, the analysts need to write their project's SRS using regular text editors and templates, such as Microsoft Word or LaTex, and the IEEE SRS template [4] respectively. There are also some tools that can produce Use Case

978-1-4244-8136-1/10/$26.00 ©2010 IEEE

descriptions and scenarios, such as DOORS [7] and STORM [3]. These tools do not generate plain natural language descriptions such as those, for example, that can be created with the use of the IEEE SRS template, and they are only applicable to Use Case modelling. Their input also has to be processed first by the analyst (the analyst has to create the use cases), while in NALASS the main input to the SRS document generation component, which is a set of formalized sentences, is automatically created by the tool, based on the user's answers to pre-determined questions. [8] describes another tool that produces an SRS document but with a very simplistic form and structure that does not cover any of the basics of the IEEE SRS template or generally the basics of good SRS documentation, including functions, sub-functions, user roles and non-functional requirements.

III. Methodology and Tool Overview

NALASS is a tool that automates the NLSSRE (Natural Language Syntax and Semantics Requirements Engineering) methodology created by [9], [10]. NALASS includes 4 components: the Formalization component for automatic creation of requirements in the form of formalized sentences, the Questions component for automatic creation of question sets to be submitted to the customer, the Diagrams component for automatic creation of diagrams, and the Documentation component for automatic generation of the SRS document. Before explaining the Documentation component in section 4, it is useful, for better understanding, to provide an overview of the methodology and the other components of the tool.

The NLSSRE methodology of [9], [10] provides formalization of the major activities of RE including Requirements Discovery, Analysis and Specification, so that the analyst will know in advance, through a step-by-step approach, what questions to ask, in what specific way to analyse the answers to the questions, and how to write them in a specific way. The application domain of the methodology is an IS (e.g. Hospital IS or Bookstore IS) that deals mainly with management of documents or other physical objects that can be conceived as electronic information which can be Created, Altered, Read and Erased.

The difference of NLSSRE with other approaches is that it provides predefined types of data and functions written as components of formalized sentences (FSRs – Formalized Sentential Requirements, the syntax of which is explained later in this section) as shown in figure 1(a), and from these sentences, it creates specific sets of questions for the users (figure 1b). The answers to these questions feed the FSRs (figure 1c) and, hence, create complete requirements (figure 1d) in the form of formalized sentences (FSRs) that can be used to create diagrammatic notations such as DFDs (fig. 2a), Class diagrams (fig. 2b) and Use Case diagrams, as well as the SRS document (fig. 6). The entire procedure is supported and automated by NALASS.

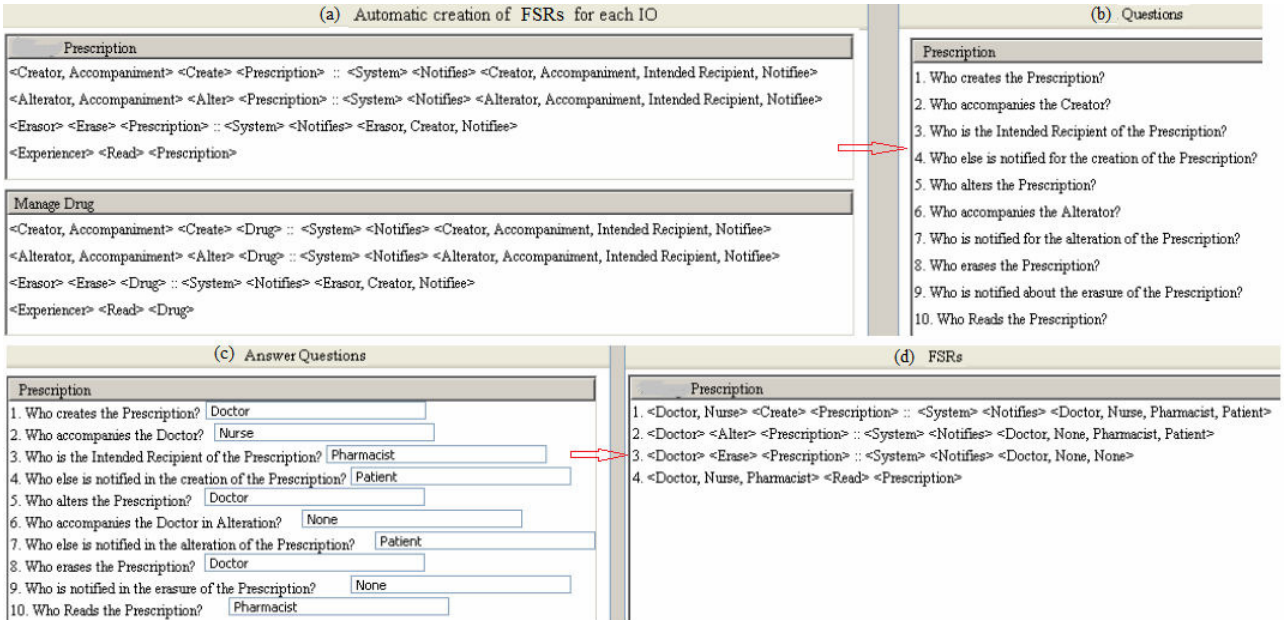

Figure 1. The predefined questions (b) created automatically by the FSRs types (a), and the resulting complete FSRs (d) created automatically by the answers of the users (c), for the *Prescription IO* – screenshots are taken from NALASS that automates and supports the NLSSRE methodology.

Apart from specificity, another advantage of NLSSRE is the use of Natural Language (NL) for the syntax of FSRs and their components. NL gives expressiveness to the formalization of requirements and makes them easily understood by the users, analysts and programmers. For example, data are derived from the semantic types of genitive case, nouns, adjectives, adverbial complements, and stable and temporary object properties; functions are derived from the semantic types of verbs; and constraints are derived from relations between data and between data and functions. And these components (functions, data and constraints) are written in the form of formalized sentences (FSRs), by using

the right order of different syntactic parts, such as subject, direct object, indirect object, etc. In this way NLSSRE provides also a common terminology for documenting data, functions and constraints. The advantage is twofold: first there will be a consistent and common language of writing, without ambiguities and redundancies, and, second, this controlled language may be computer-processed and translated automatically into diagrammatic notations and the SRS document, as already mentioned.

Another basic element of our approach is the Information Object (IO) which denotes a separate entity of information (attributes) that can stand on its own and can be created, altered, read and erased within the context of the IS. For each IO, a specific number of (FSRs) are provided. Each FSR includes a *function* (Create, Alter, Read, Erase, Notify), *non-functional requirements* (Instrument, Amount, Time, Location – due to space limitation and simplicity, they do not appear in the screenshots) with direct relation to each function, *roles* (e.g. Creator, Accompaniment) that are related to each function and are also *attributes* of the IO, and *constraints*. Hence, the FSRs facilitate the formalization of functions, data attributes, non-functional requirements and constraints of the IO. For the formalization of additional attributes of each IO, the NLSSRE methodology makes use of the genitive case, the adjective and other types of attributes (out of the scope of this paper).

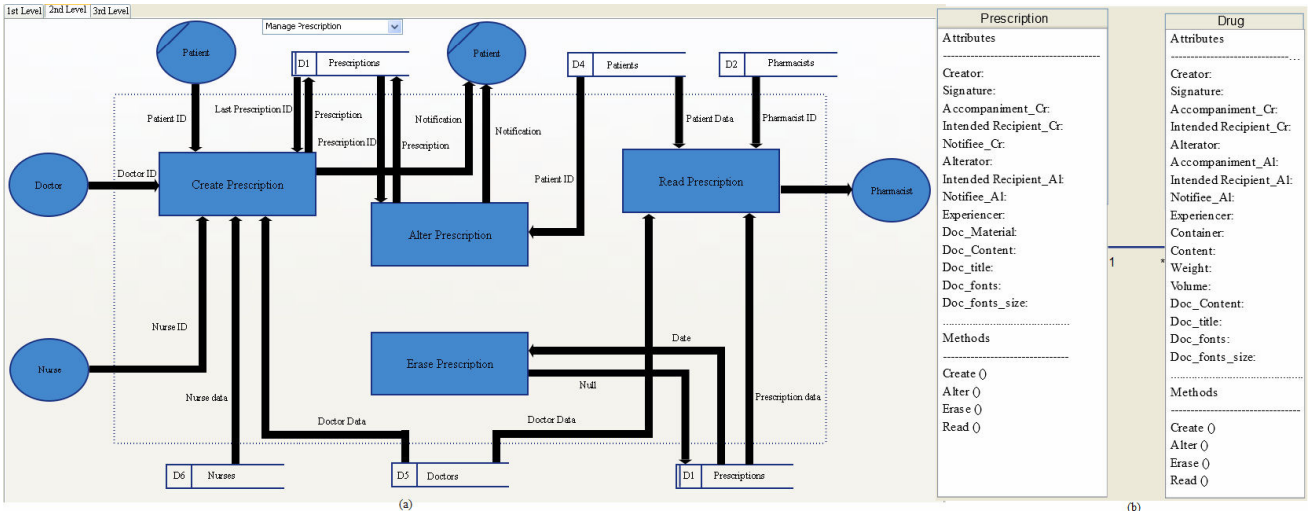

Figure 2. (a) Second Level ORDFD, (b) Class diagram

Now we illustrate with examples the two main sections of NALASS, named *Plan* and *Execution*. After creating a better understanding of these sections, we provide a detailed description of the Documentation component of the tool.

A. Plan

In the Plan section, the analyst uses a particular guide to identify and add the IOs of the IS (fig. 3). The screen in figure 3 shows some of the IOs of a Hospital Information System. Subsequently, the FSRs (fig. 1a) and the question sets of each IO (fig. 1b) are created automatically by NALASS. The questions are derived automatically from the syntactic components of each FSR; and this is the difference from most, if not all, of the approaches which use formalism in NL RE. Such approaches try to develop and formalize requirements that are already written in existing documents. We consider them not efficient, since requirements in such documents are often poorly written and organized; sentences do not necessarily follow the correct form of syntax, while there may exist redundant words, fuzzy and complicated meanings, etc. As such, it is rather precarious and difficult to apply linguistic rules on such documents.

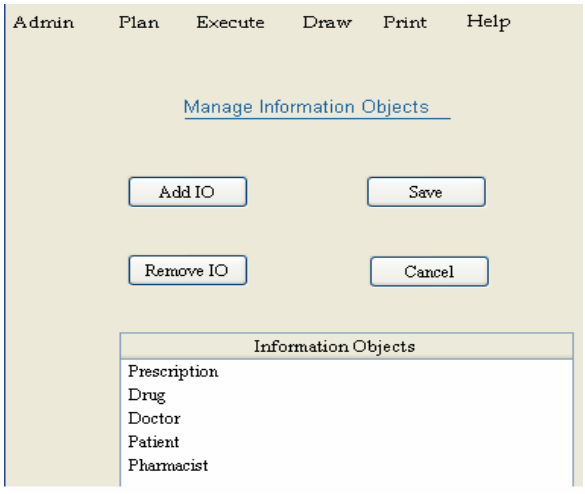

Figure 3. Adding Information Objects

B. Execution

In the 'Execution' section:

1) The analyst is in the user's environment submitting questions to the users and noting down the answers (figure 1c). The answers to the questions feed the FSRs as they are the values of the predefined types of data that comprise each

FSR, as shown in figure 1(d) (e.g. *Creator* takes the value *Doctor*).

2) The FSRs and their components, as well as other IO attributes (derived from the use of the genitive case – outside the context of this paper) are transformed to diagrammatic notations and to the SRS document, with the use of specific rules (e.g. the roles of Creator, Accompaniment, Alterator, Intended Recipient, Experiencer and Notifiee correspond to actors of a traditional DFD). Figure 2(a) shows the 2nd level DFD diagram for *Manage Prescription*, while figure 2(b) shows the class diagram for the *Prescription* and *Drug* IOs, as created by NALASS. The following section describes the SRS Document Generation component.

IV. SRS Document Generation

Figure 4 shows how the SRS document is generated by NALASS. The Documentation component receives as inputs the processed (fed with the answers of the users) elements of the NLSSRE methodology including FSRs, IOs and IO attributes, the SRS template that determines the organization and formatting of the SRS document, and the rules to convert the aforementioned inputs into a well-structured SRS document. The tool reads the template and applies: (a) *formatting rules* for the formatting of the new SRS document, by identifying the formatting elements of the template, such as fonts type and size, and line spacing, and applying them to define the format of the new SRS document; (b) *substitution* rules, by replacing the template variables included in "< >", as shown in figure 5, with the values of the components of the corresponding FSRs, IOs and attributes of IOs, as shown in figure 6 (e.g. *Information Object 1* is replaced by *Prescription*, and *Creator* of *IO 1* is replaced by *Doctor)*.

We need to mention that before the template is given as an input to the Documentation Component, it is first processed by NALASS, so as according to the given number of Information Objects, the same number of IO sections (3.1..3.n of the template) will be created. Similar checks and arrangements are done for other elements such as the types of IO attributes, which are not the same for all IOs. The way the template is built and processed does not need any further grammatical and syntactical checks, since, for example, plural is covered by the use of "(s)" at the end of the noun, the third person singular verb form is covered by the use of "(s)" at the end of each verb, and for the genitive case, we use "of" instead of "'s" for simplification and avoidance of mistakes. Future developments will include the use of a dictionary that will make such checks, thus substituting the syntactic refinements currently performed on the text as dictated by the template.

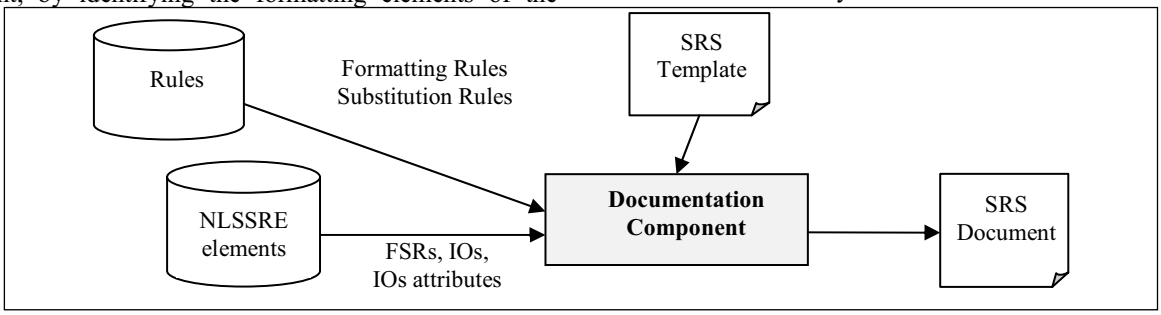

Figure 4. Configuration of NALASS's Documentation Component

As shown in figure 5, the structure of the template has been kept to essential sections of requirements specification, similar to section 3 "Specific Requirements" of the IEEE SRS document template [4]. Notably, the most substantial item in this SRS structure (sections 3 which accounts for about 70% of the SRS size) is covered in NALASS. All other items, such as the glossary of terms and the initial snapshots of the software system's user interface can be appended by the requirements analyst in the document file generated by NALASS. Future features such as use case descriptions, scenarios and diagrams will be added to the SRS document as well as DFDs and Class diagrams which are already created by the diagram component of NALASS.

NALASS can export the SRS document into either HTML or rich-text format (RTF). This will allow the requirements analyst to generate and show the specification outside NALASS's interface either in electronic or printable format. The screenshot of figure 6 shows an excerpt of the SRS document generated automatically by NALASS for the example of a new Hospital Information System.

V. Conclusions and Future Work

Research shows that there is a lack of a software tool to automate the creation of a Software Requirements Specification (SRS) document described in Natural Language and providing specific terminology for its components, including functions, data and non-functional requirements. Most of the CASE tools available focus on diagrams and pictures and cannot handle the textual aspects of requirements.

This paper has presented NALASS, a software tool that is intended to automate the application of the NLSSRE methodology which utilizes elements of natural language such as verbs, nouns, genitive case, adjectives and adverbs. Like the methodology on which it is based, the tool can be used through the entire Requirements Engineering process. In this paper we provided an overview of the methodology and the tool, and we expanded on the use of the SRS

Documentation component that generates automatically a well-structured Natural Language SRS document.

Our work is still in progress, so future considerations involve (i) automatic generation of use cases descriptions, scenarios and diagrams; (ii) embedding of DFDs and Class Diagrams (the automatic creation of which is already implemented in NALASS) to the right section of the SRS document (as also indicated in the IEEE SRS template), and (iii) assuring full consistency of the tool and the Documentation component to the methodology.

Software Requirements Specification (SRS) Template

1. Table of contents
   ……
2. Introduction
   ……
3. List of Information Objects
   The new computerized Information System includes the following Information Objects:
   <Information Object 1>
   <Information Object 2>
   ……
   <Information Object n>

3.1. <Information Object 1> Information Object
- 3.1.1. Functionality Description
  - 3.1.1.1. <Creator 1> , …., <Creator n>, <Accompaniment 1>, …, <Accompaniment n> create(s) <IO 1>. Subsequently the system notifies <Notifie 1>, …, <Notifiee n>, <Intended Recipient 1>, …, <Intended Recipient n> that <IO 1> is created.
  - 3.1.1.2. <Alterator 1> , …., <Alterator n>, <Accompaniment 1>, …, <Accompaniment n> alter(s) <IO 1>. Subsequently the system notifies <Notifie 1>, …, <Notifiee n>, <Intended Recipient 1>, …, <Intended Recipient n> that <IO 1> is altered.
  - 3.1.1.3. <Erasor 1> erases <IO 1>. Subsequently the system notifies <Erasor 1> that <IO 1> is erased.
  - 3.1.1.4. <Experiencer 1> , …., <Experiencer n> read(s) <IO 1>.
- 3.1.2. <Information Object 1> Attributes
  The following are created and compose the <Information Object 1> Information Object.
  - 3.1.2.1. Relational Attributes
    - 3.1.2.1.1. <Creator 1> ID, <Creator 2> ID, …, <Creator n> ID
    - 3.1.2.1.2. <Accompaniment 1> ID, <Accompaniment 2> ID, …, <Accompaniment n> ID
    - 3.1.2.1.3. <Notifiee 1> ID, <Notifiee 2> ID, …, <Notifiee n> ID
    - 3.1.2.1.4. <Intended Recipient 1> ID, <Intended Recipient 2> ID, …, <Intended Recipent n> ID
  - 3.1.2.2. Physical Attributes
    - 3.1.2.2.1. Width
    - 3.1.2.2.2. Length
    - 3.1.2.2.3. Material
    - 3.1.2.2.4. ……
  - 3.1.2.3. Document Attributes
    - 3.1.2.3.1. Title
    - 3.1.2.3.2. Content
    - 3.1.2.3.3. Fonts {type, size,…}
    - 3.1.2.3.4. ……
- 3.1.3. Functions
  - 3.1.3.1. Create <Information Object 1>
    - 3.1.3.1.1. Description: The System compares initial candidate values (ICVs) of each attribute of <Information Object 1> to relevant constraints. Subsequently the system adds the initial candidate value of each attribute of <Information Object 1>.
    - 3.1.3.1.2. Sub-Functions of Create <Information Object 1>
      - 3.1.3.1.2.1. Compare <ICV 1> for <Attribute 1> of <IO 1> to <Constraint 1: User must select from a list of constraints>
      - 3.1.3.1.2.2. Add <ICV 1> for <Attribute 1> of <IO 1>
    - 3.1.3.1.3. Roles of Create <Information Object 1>
      - 3.1.3.1.3.1. <Creator 1>, …, <Creator n> is/are the creator(s) of <Information Object 1> and responsible for the content of <Information Object 1>
      - 3.1.3.1.3.2. <Accompaniment 1>, …, <Accompaniment n> help(s) the <Creator 1>, …, <Creator n> to create <Information Object 1>.
      - 3.1.3.1.3.3. <Intended Recipient 1>, …, <Intended Recipient n> is/are the intended recipient(s) of <Information Object 1>. <Intended Recipient 1>, …, <Intended Recipient n> must Read prescription and take appropriate actions (see Read <Information Object 1>).
      - 3.1.3.1.3.4. <Notifiee 1>, …, <Notifiee n> is/are notified to communicate with <Intended Recipient 1>, …, <Intended Recipient n>
    - 3.1.3.1.4. Non-functional requirements for Create <Information Object 1>
      - 3.1.3.1.4.1. Instrument used to create <Information Object 1>
      - 3.1.3.1.4.2. Amount of time needed to complete Create <Information Object 1>
      - 3.1.3.1.4.3. Location
      - 3.1.3.1.4.4. Time occurred
      - 3.1.3.1.4.5. ……
  - 3.1.3.2. Read <Information Object 1>
    - 3.1.3.2.1. Description
    - 3.1.3.2.2. Sub-functions of Read <Information Object 1>
    - 3.1.3.2.3. Roles of Read <Information Object 1>
    - 3.1.3.2.4. Non-functional requirements for Create <Information Object 2>
  - 3.1.3.3. Alter <Information Object 1>
  - 3.1.3.4. Erase <Information Object 1>

3.2. <Information Object 2> Information Object

3.3. ……

3.n. <Information Object n> Information Object

4. Initial snapshots of the user interface
5. Glossary
6. Appendices

Figure 5. SRS Document Template given as input to NALASS

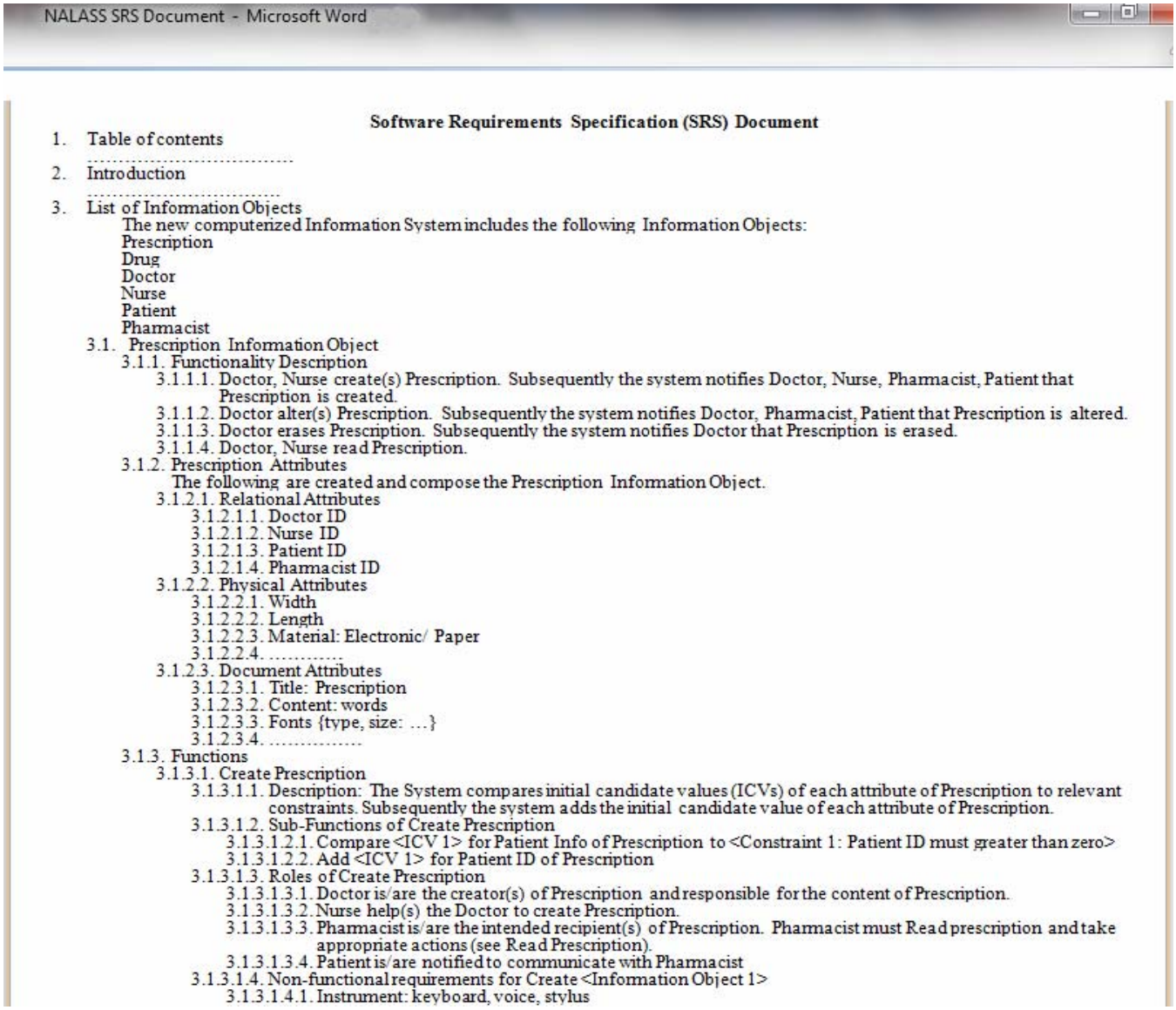

Figure 6. Excerpt of the SRS Document created automatically by NALASS

REFERENCES

[1] D. S. Le Vie, "Documentation Metrics: What do YouReally Want to Measure?," STC Intercom, pp. 7-9, 2000.

[2] S.L. Pfleeger, and J.M. Atlee, Software Engineering: Theory and Practice. Upper Saddle River, New Jersey, USA, Prentice Hall, 2006.

[3] S. Dascalu, E. Fritzinger, K. Cooper, and N. Debnath, "A Software Tool for Requirements Specification: on Using the STORM Environment to Create SRS Documents", Proc. of the Second International Conference on Software and Data Technologies (ICSOFT-2007), July 2007, pp. 319-326.

[4] IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications, IEEE Xplore, 1998.

[5] IBM Rational Rose. Available online as of July 12, 2010 at http://www-306.ibm.com/software/rational/

[6] MagicDraw. Available online as of July 12, 2010 at http://www.magicdraw.com/

[7] DOORS. Telelogic's DOORS. Requirements management traceability solutions. Available online as of March 31, 2007 at http://www.telelogic.com/products/doorsers/index.cfm

[8] N. Kassel and B.A. Malloy, "An Approach to Automate Requirements Elicitation and Specification," Proc. of the 7th Int. Conf. on Software Engineering and Applications, November 3-5, 2003, Marina del Rey, CA, USA, pp. 544-549.

[9] M. Georgiades, A. Andreou, and C. Pattichis, "A Requirements Engineering Methodology based on Natural Language Syntax and Semantics," Proc. of the 13th IEEE International Requirements Engineering Conference (RE'05), August 29-September 02, 2005, Paris, France, pp. 473-474.

[10] M. Georgiades, A. Andreou, "A Novel Methodology to Formalize the Requirements Engineering Process with the Use of Natural Language," Proc. of the IADIS Intern. Conf. on Applied Computing, October 14-16, 2010, Timisoara, in press.