Natural Language and Requirements Engineering — Nu?

Daniel M. Berry
CSD & SE Program
University of Waterloo

Natural Language and Requirements Engineering

Daniel M. Berry
CSD & SE Program
University of Waterloo

Outline of Talk

- **Introduction**
- **Story of the Title**
- **Overview, in Rough Chronological Order**
- **Lessons Learned**
- **Specifics**
  - **Kevin Ryan's Observations on AI in RE**
  - **Abstraction Identification**
  - **Ambiguity**
  - **Understanding NL**
  - **Improving RS Writing**

Introduction

Natural Language (NL) Requirements Specifications (RSs):

Overwhelming majority of RSs are written in NLs.

Virtually every initial conception for a system is written in NL.

Virtually every RFP is written in NL.

But...

We all know that...

NL is *so* imprecise, so ambiguous, and so inherently so!

No wonder RSs are such messes!

There is that old tradeoff: RSs written in

- **NLs or**
- **Mathematics-Based (MB) Formal Languages (FLs)**

NL:

– **inherently ambiguous**

\+ **always someone who can write it**

\+ **always more or less understood by all stakeholders,**
albeit somewhat differently by each

MB FL:

+ inherently unambiguous

- not always someone who can write it

- not understood by most stakeholders, although all that do understand it understand it the same

Focus of Research

A lot of research in RE is directed at solving the problem of imprecise ambiguous RSs by

- **convincing people to use MB FLs and**
- **addressing the negatives of FLs**

...

Addressing Negatives of FLs

by making FLs generally more accessible by

- **coming up with simpler FLs, e.g.**
  - **Parnas's tabular form**
- **building tools that take the drudgery out of using FLs, e.g.,**
  - **smart editors**
  - **theorem provers**
  - **model checkers**
  - **code generators**

Addressing Negatives of FLs, Cont'd

- **building tools that help people understand FL specifications, e.g.,**
  - **simulators**

Stark Reality

But, the reality is that there is no escaping NL RSs.

Michael Jackson [Jackson1995] reminds us that

> **Requirements engineering is where the informal meets the formal.**

Stark Reality, Cont'd

Therefore, NLs are inevitable, even if it is only for the initial conception.

(Unless the client is some really weird math-type nerd who thinks in first-order predicate calculus.)

Stark Reality, Cont'd

Even if one moves immediately to FLs, the inherent ambiguity of the NL initial conception can strike as the transition is made.

What the formalizer understands of the conception may be different from what the conceiver meant.

Semi-Formal Languages?

What about semi-formal languages like UML models?

Double whammy

Ambiguity can still strike when go from conception to model.

And

The model itself is not unambiguous.

Research in NLs in RE

Therefore, there is a group of researchers focusing on solving the problem of imprecise, ambiguous RSs by trying to improve our writing, understanding, and processing of NLs.

There are several approaches to avoiding the ambiguity and imprecision of NLs:

Problem Avoiding Approaches:

1. **Learn to write less ambiguously and less imprecisely.**
2. **Learn to detect ambiguity and imprecision.**
3. **Use a restricted NL which is inherently unambiguous and more precise.**

The first two reduce the disadvantage of existing NLs.

The third restricts the NLs to disallow the disadvantages.

Direction of this Talk

This talk surveys this research.

Some of it is by me and people with whom I work; I know it well and can represent it well.

Some of it is by others; I do not know it as well and might not accurately represent it. For this, I apologize in advance.

Story of the Title

The web page for this workshop shows:

Dan Berry, University of Waterloo, Canada
Natural language and requirements engineering - nu?

Abstract. I argue that RE is where the formal meets the informal. I argue that it is inevitable that natural language be used for requirements. I discuss the the tradeoff between natural language and formal languages in ambiguity and readability. I argue that the traditional approach of solving this problem by trying to get everybody to be formal is doomed, that we should focus instead on doing a better job in natural languages.

Where did this come from?

I got a message from Martin, on behalf of himself and Roel, the chairs of this workshop (a line with “>” in column 1 is from the original message and
a line without that is from my answer):

>Hi Dan,

>as you know, Roel Wieringa and I are co-chairing a Requirements
>Engineering Workshop that will take place immediately after the RE’01
>PC meeting in London.
Yes.. I know..

>We plan a discussion-oriented workshop with four sessions of 90
>minutes each:

> 9:00 - 10:30 Session 1: Problem frames in requirements engineering
>11:00 - 12:30 Session 2: Software requirements and systems engineering
>13:30 - 15:00 Session 3: Abstraction levels in requirements engineering
>15:30 - 17:00 Session 4: Natural language and requirements engineering

Good selection of topics.. should be exciting too.. well done..

(skipping)

>We would like to invite you as a speaker for session 4 on NL and RE.
>The other speaker will be Didar Zowghi from University of Technology,
>Sydney, Australia.
a worthy colleague..

>Could you please confirm asap if you accept our invitation and give
I accept your invitation..

>us a working title and/or some keywords of your talk? The final title

How about "Natural Language and Requirements Engineering"
Nu?

My “Nu?” was in reference to the fact that I used the title of the session as the title of my talk; as if to say “What else did you expect?”

In any case, Martin did not quite understand and, as a good requirements engineer, he asked me:

>What does "Nu?" mean?
Well.. it's a Yiddish expression stolen from Russian that means "well?" (the two "wells" in the above sentence have the same meaning but the first is indicative and the second is a question..)

If you recall.. you told me the title of the session and then you asked me for a title of my presentation.. so I, for the lack of anything better, chose as the title of my presentation.. the title of the session.. and to that idea I added the exclamation "Nu?" as if to say.. "Well.. what else did you expect..".. (if you could see me, you would see my shoulder's shrugging and my hands cupped in front of me as if I am asking a question..)

I guess it's hard to explain in writing .. but it would be clearer if you could hear my voice and see me.. OR you were familiar with Yiddish humor.. Nu? (and that was a correct usage of the term right there!!)

:-)

The fact that my meta remark “Nu?” ended up in the final title is a perfect introduction to a key point of this talk, about ambiguity in natural language specifications, Nu?.

Later, Colin Potts replaced Didar Zowghi as the other speaker.

He sent me a message in which he used “nu?” as meta comment:

>I am now on the programme after you talking about “Everything a
>Requirements Engineer Always Wanted to Know about Cognitive Linguistics
>(But Was Afraid to Ask)” Pretty good title, nu? You can judge the
>content in a couple of weeks!

I wonder where he got the “nu?” from?

BTW, I replied to Colin, simply, “Nu!”.

Overview

In Rough Chronological Order:

- **Information Retrieval**
- **Processing Restricted NL RSs**
- **Unrestricted NLs in Analysis and Design**
- **Extracting Information from Grammatical Structure**
- **AI for RE?**
- **Abstraction Identification**
- **Model Building from Unrestricted NL RSs**
- **Avoiding or Detecting Pitfalls in NL RSs**

Information Retrieval (IR)

Particularly indexing, keyword identification, and thesaurus generation

Mostly batch processing on large collections of documents, e.g., for Library of Congress

The work goes way, way back, but is summarized by [Salton1989, Frakes1992, Srinivasan1992].

Processing Restricted NL RSs

By restricting the NL (but then it is not so natural!), it becomes possible to ascribe precise semantics to the sentences.

However, the questions always arise:

- **How expressible is the language? Have we lost something valuable in the restrictions?**
- **How convenient is it for humans to write the language?**
- **Is writing in such a language really different from writing in MB FLs?**

James Comer's Early Work

James Comer described an interactive system converting descriptions of data structures given in a restricted subset of English into algebraic axiomatic data type specifications [Comer1983].

TELL

This was followed by other efforts, e.g., Hajime Enomoto *et al* developed TELL.

TELL translates specifications written in restricted NL, NSL, into formulae of modal logic [Enomoto1984b].

TELL is used to specify a network protocol that is then verified to be the alternating bit protocol [Enomoto1984a]

Attempto

Norbert Fuchs *et al* have developed Attempto, a restricted English and a tool for understanding it [Fuchs1999].

It has been used for a variety of medium-sized examples and as the input language for a theorem prover.

Unrestricted NL in Analysis and Design

Russ Abbott described the use of unrestricted natural language as part of an object-oriented requirements analysis and program design method [Abbott1983].

This approach became a key element of Grady Booch's approach to object-oriented program design [Booch1991].

Extracting Information from Grammatical Structure

Motoshi Saeki, Hisayuki Horai, *et al* consider what semantics they can learn from the syntax structures of unrestricted NL RSs [Saeki1987].

Grammatical Structure, Cont'd

Colette Rolland and Christophe Proix explored the relationship between grammatical cases of words in NL RSs and ER models and use what they learned to build a tool, OICSI [Rolland1992].

These and other similar approaches are summarized in a survey [Denger2001].

AI for RE?

Kevin Ryan wondered if AI approaches to NL understanding had any chance of working for RE [Ryan1993].

His conclusion was basically, “No!”.

Abstraction Identification

Starting from the IR work, a series of approaches improved the means for tool-assisted identification of abstractions in NL problem descriptions and RSs.

AbstId, Cont'd

These include using

- **parsers [Berry1987]**
- **repeated phrase finders [Aguilera1990]**
- **lexical affinity finders [Maarek1989]**
- **signal processing techniques [Goldin1997]**
- **document comparison [Lecœuche2000]**

AbstId, Cont'd

The last two combined seem to do the whole job.

Signal processing finds the important concepts that are repeated frequently.

Comparison finds the important concepts that are mentioned only rarely.

Model Building from Unrestricted NL RSs

As mentioned above, Rolland and Croix built a tool that builds ER models from the grammatical case tagging given to nouns in a NL RS [Rolland1992]

Julio Leite and Ana Paula Franco considered a technique for deriving a conceptual model from a lexicon, which is in turn derived from NL RSs [Leite1993].

Model Building, Cont'd

Yasunori Ishihara *et al* developed a tool that translates NL specifications of network protocols into algebraic specifications [Ishihara1993].

Stephane Somé *et al* developed a tool for elicitation of scenarios written in NL and then to derive complete RSs from them [Somé1996].

Gervasi's Work

Most recently, Vincenzo Gervasi has developed CIRCE, a complete environment for interactive parsing and understanding NL RSs and for deriving a number of models of the specified system [Gervasi2000a, Ambriola2000].

He and Bashar Nuseibeh have applied CIRCE to an industrial strength example, i.e., the International Space Station's Node Control System [Gervasi2000b].

Avoiding or Detecting Pitfalls in NL RSs

Some are focusing on the actual writing of the NL RS by the human writers, with an aim that they produce more precise, complete, consistent, and less ambiguous NL RSs.

Some are focusing on improving inspections of NL RSs to more reliably detect incompleteness, inconsistencies and ambiguities.

Dangerous Constructions

Christine Rupp and Rolf Götz detail common problems in NL RSs that lead to ambiguities, incompletenesses, and inconsistencies [Rupp1997].

- **Writers can use this list to avoid problems.**
- **Inspectors can use this list as a checklist in inspections.**

Dangerous “All”

Erik Kamsties and I have sharpened one of these items, the recommendation that universal quantifier equivalents, e.g., “all”, “never”, etc. are often wrong [Berry2000].

Guide for Writing NL RSs

There is at least one book out there on the writing of NL RSs, by Benjamin Kovitz [Kovitz1998].

Contracts and RSs

Legal contracts and software RSs are similar in that

- **both are written in NLs and**
- **both have to anticipate all possible contingencies.**

Contracts and RSs, Cont'd

A team was formed of two lawyers and two requirement engineers.

- **Mickey Krieger, a lawyer who is a mathematician and computer scientist;**
- **David Kay, a lawyer who is a software engineer; and**
- **Kamsties and I, two software engineers who are requirements engineers.**

Writing Unambiguously

This lawyers–requirement-engineers team has written a handbook on writing unambiguous NL legal contracts and software RSs [Berry2001].

Writing Unambiguously, Cont'd

The handbook focuses on ambiguities stemming from

- **incorrect placement of "only" and words like it, and**
- **incorrect uses of**
  - **grammatical number**
  - **logical connectives**
  - **universal and existential quantifier equivalents, etc.**

Surfacing Ambiguity

Finally, Kamsties has studied ambiguity in NL RSs and has come up with experimentally validated methods to detect them in inspections [Kamsties2001a, Kamsties2001b].

Lessons Learned

- **Language Understanding**
- **Abstraction Identification (AbstId)**
- **NLs for Requirements Documents**
- **Ambiguity**

Language Understanding

- **AI for RE**
- **Too Much Automation is Bad for RE**
- **RS Text vs. Ordinary Text**

AI for RE

Artificial intelligence approaches to language understanding do not work well enough for RE work.

The mistakes are annoying.

Too much good stuff is missed.

AI for RE, Cont'd

We prefer naturally dumb clericality (פקידותיות (picky-duty-ut)) to artificial intelligence.

Stupidity is preferred to intelligence if the latter can lose information as a result of it not ever being perfect.

Too Much Automation is Bad for RE

I regard with suspicion any tool that is so automatic that it takes the human requirement engineer out of the loop.

The most powerful RE tool ever is the good ol' human brain.

Tools that require human intervention bring the human into the loop and promote thinking about the results.

Too Much Automation, Cont'd

It is in this thinking that omissions and questionable, albeit logically okay, requirements are noticed.

Tools that run by themselves to produce requirements take the human out of the loop and make it less likely that a human will think about the results.

This lack of thinking is very, very dangerous.

RS Text vs. Ordinary Text

The NL used in RSs is quite different from the NL used in ordinary, garden variety or even highly literary text.

RS text is highly explicit, over a restricted domain, using a reduced vocabulary, with well-defined terms.

The other kind of text is much more elliptical, over arbitrary domains, using an unlimited vocabulary, with fuzzily defined terms.

AbstId

- **grep vs. Eyes**
- **Requirements for AbstId**
- **Differences between IR and RE**
- **Kinds of Searching for AbstId**

grep vs. Eyes

***grep* is a lot better at finding all instances of a word or a pattern than is the human eye.**

Requirements for AbstId

The total amount of information to deal with for any real problem is HUGE and repetititive.

We desire assistance in extracting useful information from this mass of information.

Requirements for AbstId, Cont'd

We would like the extracted information to be

- **summarizing (got less stuff),**
- **meaningful (precision) (got only good stuff), and**
- **covering (recall) (got all good stuff).**

From 500 pages, we want 5 pages containing *all* and *only* the meaningful information in the 500 pages.

Requirements for AbstId, Cont'd

We prefer less summarization and occasional meaningless stuff than to lose some meaningful stuff, because in any case, a human will have to read the output and at that time can filter out the meaningless stuff.

Differences between IR and RE

Both IR and RE deal with extracting small pieces of information from large texts.

However, there are key differences between IR and RE that impact the requirements for tools used for their extraction processes.

Differences, Cont'd

IR tends to work with continually growing large collections of unchanging texts.

RE works with small collections of rapidly changing texts.

Thus, rapid, off-line, batch processing is essential for IR.

Thus, slower, interactive, on-line processing is acceptable for RE.

Differences, Cont'd

The tools for IR cannot make mistakes. They must be summarizing, meaningful (precise), and covering (recalling) without any human intervention.

The tools for RE can afford to be less than meaningful (precise) because a human being is watching the output and can filter meaningless stuff, provided that there is not too much of it.

Differences, Cont'd

In RE, you can err on the side of meaningfulness (precision) provided that you lose no coverage (recall).

In RE, it doesn't matter if it takes 3 hours to extract desired information; 3 hours is meaningless in the lifecycle; you can take the client out for lunch during the 3 hours.

Differences, Cont'd

These differences affect requirements and implementation choices for tools for IR and RE.

We cannot use standard IR tools for extraction in RE without rethinking their applicability.

Kinds of Searching for AbstId

In abstraction identification, we need to do both frequency based searching and comparison based searching:

- **Frequency based searching finds the important concepts that are repeated more often than a threshold amount.**

Kinds of Searching, Cont'd

- **Comparison-based searching, in which a domain-specific document is compared with a vanilla, general document, finds the important concepts that are mentioned rarely, including individual synonyms; it also filters out noise.**

In these searches, concepts are potentially non-contiguous, potentially permuted phrases, made of words or word fragments.

NLs for Requirements Documents

- **NLs for Scenarios**
- **NLs for Users' Manuals**

NL for Scenarios

Expressing scenarios in natural language keeps them understandable by the clients and users but runs the risk of ambiguity.

In particular, a user can immediately see that a proposed scenario does not capture what he or she would do.

This is much harder to see in a feature centered, functional SRS, whether formal or informal.

NL for Users' Manuals

The user's manual, written in natural language, turns out to be a very good requirements specification, ...

particularly if it is based on use cases and scenarios.

Ambiguity

- **Why Ambiguity is Tough to Find**
- **Not All Ambiguity is Linguistic**
- **Poor Writing Causes Ambiguities**

Why Ambiguity is Tough to Find

Many ambiguities are not noticed because of subconscious disambiguation.

The reader understands an interpretation and thinks that it is the only one.

This is why an inspection checklist with one ambiguity relevant item, “Is document ambiguous?” is ludicrous.

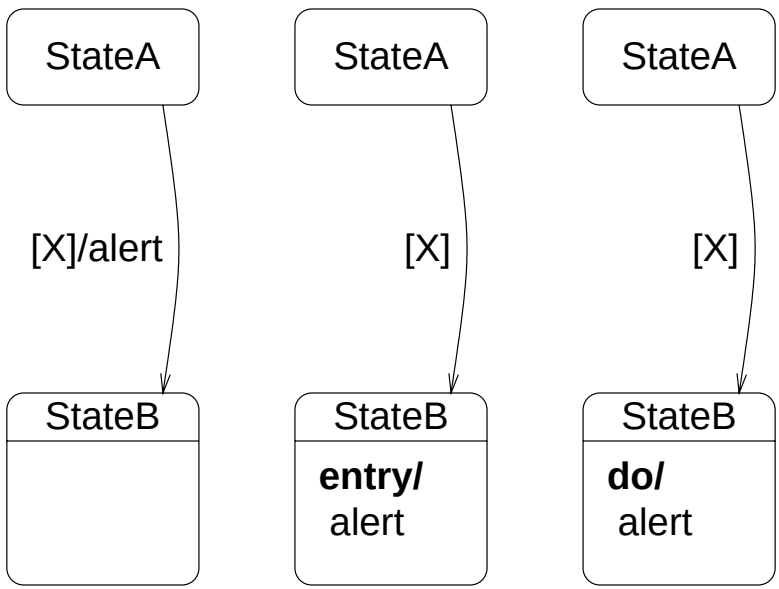
Not All Ambiguity is Linguistic

Some ambiguities are due specifically to computing and do not appear ambiguous from the linguistic point of view, e.g.

"If X happens then raise an alert."

is clear to the reader (assuming that X **is clear), but gives rise to three different UML State Machine models, each with slightly different semantics.**

Not All Ambiguity, Cont'd



At implementation and run times, which one is meant can be critical.

Poor Writing Causes Ambiguities

People write appallingly poorly and ambiguously when it comes to words like

- **only**
- **all**
- **each**
- **not | no | none |**
- **pronouns**

Poor Writing, Cont'd

These people include even lawyers and requirements engineers who should know better, because their documents have a heavy requirement for lack of ambiguity!

Even more appalling is the fact that neither law degree nor SE degree programs teach people how to write clearly.

Poor Writing, Cont'd

Each program focuses on using its field's technology to solve problems to which poor writing is a major contributor.

It's like teaching SE without teaching RE, because it's in poor RE that about 80% of the errors are introduced.

Specifics

- **Kevin Ryan's Observations on AI in RE**
- **Abstraction Identification (AbstId)**
- **Ambiguity**
- **Understanding NL**
  - **Restricted NL**
  - **Unrestricted NL**
- **Improving RS Writing**

Kevin Ryan's Observations on AI in RE

Back in 1993, Kevin Ryan put the role of NLP in RE into perspective [Ryan1993].

He noted the importance of NL to RE, that indeed RE is firmly based in NL, as spoken and written by humans.

He observed:

Kevin Ryan's Observations, Cont'd

The history of natural language processing (NLP), in relation to the specification of systems and programs has been bedevilled with many unrealistic suppositions and presumptions. Given the critical and expensive nature of the current approaches to requirements engineering (RE) the prospect of a support system that would automatically understand a user's needs is, naturally, very appealing. Numerous research projects have proposed to derive and validate system requirements knowledge by means of a natural or "near natural" conversation with the prospective client [e.g. *citations*].

Kevin Ryan's Observations, Cont'd

This facility, it is fondly believed, is both feasible and desirable and would make specification of systems both easier and more accurate. Unfortunately this belief is incorrect on both counts. In this brief paper, I wish to assert that natural language processing does not now, nor will it in the forseeable future, provide a level of understanding that could be relied upon, and even if it could, it is highly questionable that the resulting system would be of great use in requirements engineering.

Kevin Ryan's Observations, Cont'd

I would add that such a system would not even be desirable, because it would take the thinking requirements analyst out of the loop, making it less likely that he or she would notice serious omissions and questionable, albeit logically okay, requirements.

While full understanding is out of the realm of possibility or desires, there are ways that NLP tools can help the practicing, thinking requirements analyst.

Kevin Ryan's Observations, Cont'd

Such tools can help scan, search, browse, and tag early textual descriptions of the requirements to assist in the analysis and social processes that are necessary to produce full and accurate requirements specification.

Such tools can help maintain fully traced requirement specifications throughout the full lifecycle of the required system.

AbstId

On surface, AbstId is similar to the keyword identification or the back-of-book indexing (KWId/Ind) problem, in which keywords or index terms are extracted from a full text.

Important Measures:

***Recall*:**

proportion of relevant material that is retrieved

Did it get the good stuff? Is it covering?

***Precision*:**

proportion of retrieved material that is relevant

Did it get only the good stuff? Is it all meaningful?

Important Measures, Cont'd

***Summarization*:**

proportion of material that is retrieved

Did it successfully reduce what needs to be examined closer?

Important Measures, Cont'd

Note that the original document has 100% recall.

A document that is 100% precise may still repeat each item a lot.

So, you want good summarization also.

From 500 pages, we want 5 pages containing *all* and *only* the meaningful information in the 500 pages.

AbstId and KWId/Ind Differences

But, there are some key differences between AbstId and KWId/Ind:

AbstId	KWId/Ind
One document at a time	Mass indexing of new documents
Interactive	Batch
May be imprecise if recall is perfect	Must have perfect recall and precision
May be fooled	Must not be fooled

(“May” = “It is acceptable for it to be”, because interaction allows human to notice problem, if the software has not.)

The History of AbstId

1. **Abbott and Booch**
2. **Berry**
3. **Aguilera**
4. **Synonyms and Stop Words**
5. **Maarek**
6. **Goldin**
7. **Lecœuche**
8. **Other Work**

Abbott and Booch

Abbott and Booch [Abbott1983, Booch1991] suggest identification of nouns (= type and object abstractions) as part of OO design and programming.

However, the identification is entirely manual.

Easy for human to overlook a lot.

High precision, low recall.

Berry

Berry used UNIX's *style -P* (parts of speech finder) to find the nouns [Berry1987].

Too many distracting, and sometimes funny, mistakes.

Low precision, low recall.

Aguilera

Aguilera used *findphrases*, a repeated phrase finder developed for back-of-the-book indexing [Aguilera1990].

Found only repeated contiguous phrases.

Medium precision, medium recall

Synonyms and Stop Words

All of above and below are improved by use of synonym replacement and elimination of stop words (“the”, “and”, etc.)

Maarek

Maarek used lexical affinities instead of repeated phrases [Maarek1989].

Two words are in a lexical affinity if they appear often in the same order not too far apart from each other.

Found repeated non-contiguous phrases, but only with words in the same order each time.

Medium precision, medium to high recall

Goldin

Goldin used a variation of signal processing.

goldin berry abstfinder automated software

Sentence is a stream of bytes (not words; blank just a character).

Goldin, Cont'd

Compare each one-character circular shift of a sentence with all circular shifts of all sentences and count matching positions; the matching positions themselves are the identified abstractions which may straddle word boundaries.

Finds frequent multicharacter abstractions which may be split and in different orders; misses low frequency abstractions.

Goldin, Cont'd

Medium precision, fairly high recall.

In an industrial strength RFP, Goldin, a nonexpert in domain, with the help of *AbstFinder* found in 5 hours all and a few more abstractions than did 3 domain experts working full time for a month.

Lecœuche

Up to now, everyone misses infrequent terms

Lecoeuche compares document of interest with another, vanilla document, reporting terms that do not show up in both [Lecœuche2000].

This finds the domain-specific terms that are used only once, e.g., an important technical term used once in a definition, because it is not used in the vanilla document.

Lecœuche, Cont'd

No need to eliminate stop words ahead of time; they show up in abundance in both.

Even reduces need for synonym replacement; each different term is used rarely compared to a common replacement.

When combined with *AbstFinder*, you get very high recall. Still only medium precision; but the human operator filters out junk.

Other Work

Closely Related Work: Extracting Concepts

Extracting concepts from file names using clustering [Anquetil1998]

Disciplined natural naming, a systematic method to make concept, model, procedure, and file names in software development [Bowden1998]

Other Work, Cont'd

Automatic acronym extraction by a knowledge elicitation program [Laitinen1992]

Ambiguity

We consider

- **what is wrong with ambiguity and**
- **avoiding or detecting ambiguity**

What is Wrong with Ambiguity?

Ambiguity in requirements specification is dangerous because stakeholders (customers, users, developers) can disagree on the meaning of a requirements specification without being aware of the disagreement.

This disagreement can result in disastrous failures as the software is not prepared to deal with expected situations or deals with them in unexpected ways.

What is Wrong?, Cont'd

On the other hand, ambiguity can be quite acceptable provided that contextual information is available, because humans are naturally skilled in resolving ambiguities, sometimes without even knowing it.

Often disambiguation is subconscious and tacit, as the reader of an ambiguous phrase is not even aware that there is an interpretation other than the one that came first to his or her mind.

Avoiding or Detecting Ambiguity

We are reducing the disadvantages of existing NLs by

- **helping the specification writer to avoid ambiguities, and**
- **helping the specification reviewer to detect ambiguities.**

Surfacing Ambiguities

Erik Kamsties recently completed his PhD dissertation, *Surfacing Ambiguity in Natural Language Requirements* [Kamsties2001b].

Kamsties observes that ambiguity is more complex than is often recognized in RE literature.

Surfacing Ambiguities, Cont'd

The emphasis of his work is on reducing ambiguity during modeling and analysis of the requirements, instead of only trying to identify them later only after the requirements are specified.

Surfacing Ambiguities, Cont'd

Kamsties suggests techniques, based on industrially proven inspection and modeling techniques, that enable a requirements engineer to spot dangerous ambiguities, i.e., ones that are misinterpreted.

1. **Checklists**
2. **Modeling**
3. **Scenario-Based Reading**

Checklists

The core of the checklist approach is a comprehensive set of definitions of both kinds of ambiguity that occur in NL RSs:

- **linguistic (syntactic and semantic) ambiguity and**

- **conceptual (requirements and computing relevant) ambiguity.**

Checklists, Cont'd

The definition of linguistic ambiguity was borrowed from linguistics; nu?.

These definitions of ambiguity can be used as a checklist for inspections of NL RSs.

Modeling

He developed a procedure for deriving ambiguity-surfacing guidelines from an arbitrary modeling technique to be applied during application of the technique.

Modeling, Cont'd

He used this procedure to develop ambiguity surfacing guidelines that are tailored to the modeling techniques,

- **for SCR (Software Cost Reduction) and**
- **for UML (Unified Modeling Language)**

Modeling, Cont'd

The guidelines are normally applied during the modeling of requirements.

These guidelines are adaptable for use also in inspections.

Scenario-Based Reading

The scenario-based reading techniques, based on ideas of Victor Basili [Basili1997]

provides the inspector with an operational scenario (of something to do to the specification) that requires him or her to create an abstraction of the requirements and then to answer questions based on an analysis of the abstraction, e.g.

Scenario-Based Reading, Cont'd

- **abstraction = test cases**

  question = "Do you have all the information needed to develop a test case?"

  and

- **abstraction = matrix of interacting requirements**

  question = "Is there only one relation between requirements A and B?"

Experimental Validation

He experimentally validated and compared the effectiveness of the techniques in controlled experiments at Fraunhofer IESE with SE students and professionals on problems of industrial size.

Experimental Validation, Cont'd

These techniques led to

- **reduced numbers of ambiguities misinterpreted during requirements modeling and**

- **higher numbers of detected ambiguities in inspections.**

over previous unfocused techniques.

Experimental Validation, Cont'd

Moreover, he was able to rank the techniques according to effectiveness.

1. **Scenario-Based Reading**
2. **Modeling**
3. **Checklists**

Understanding NL

Because of the slow progress of language understanding work in AI, the initial work in language understanding for RE was with restricted languages.

Restricted NL

In a 1984 publication Enomoto, Yonezaki, Saeki, Chiba, Takizuka, and Yokoi describe a tool TELL which accepts specifications written in a restricted natural language NSL. It translates these descriptions into formulae of modal logic so that a rigorous machine-processable semantics is determined [Enomoto1984b].

Restricted NL, Cont'd

They used the TELL system to specify a network protocol that is verified to be the alternating bit protocol [Enomoto1984a]

In 1999, Fuchs, Schwertel, and Schwitter issued the third version of Attempto Controlled English (ACE), a system specification language [Fuchs1999].

Restricted NL, Cont'd

ACE is a controlled NL, a subset of English with a domain-specific vocabulary, a restricted grammar, and precisely defined semantics. ACE has been used to specify a simple automatic teller machine, Kemmerer's library data base problem, Schubert's steamroller, and has been used as the input language of a theorem prover.

Unrestricted NL

In the late 1990s, Vincenzo Gervasi began to explore ways to understand unrestricted NL RSs.

He took advantage of peculiarities of processing NL text for requirements, properties that are not true in ordinary, general text.

Unrestricted NL, Cont'd

NLP for Requirements	NLP for Arbitrary Text
Explicitness is desired	Elliptical, depends on context
Mostly interactive redaction	Often batch processing
Restricted domain	Arbitrary domain
Reduced vocabulary	Vast vocabulary
Well-defined terms	Mostly approximate definitions

Unrestricted NL, Cont'd

These differences come from three main factors:

1. **Explicitness — tacit information is made explicit**
2. **Interactivity — the requirements engineer oversees redaction**
3. **Repetititititiveness — domain knowledge is reused and repeated**

CIRCE

Gervasi implemented an environment, CIRCE, for support of NL requirements writing and analysis:

CIRCE, Cont'd

CIRCE helps the requirements engineer produce

- **document models**
- **system models**
- **process models**

from a set of NL requirements, helping him or her

- **to validate those models and**
- **to measure their characteristics.**

CIRCE, Cont'd

He has done two major case studies to validate the effectiveness of his approach

- **International Space Station's Node Control System**
- **Universität Kaiserslautern's Light Control System for Building 32.**

CIRCE, Cont'd

CIRCE includes CICO, a simple NL parser.

CICO is shallow parser for NL implementing domain-based parsing based on a fuzzy rewriting system.

Shallow parsing works for RSs since explicitness promotes adherence of surface structure of sentences to semantic content.

Shallow parsing is efficient.

CIRCE, Cont'd

Fuzzy matching improves the robustness of shallow parsing, so that it can succeed even when parsing cannot match a sentence exactly.

Backtracking and heuristic optimization are used to determine an optimal parse tree for a statement.

Thus, the parse is only approximate, but good enough.

CIRCE, Cont'd

Domain-specific rules encourage the use of standard language, vocabulary, and style.

These rules include a synonym glossary.

So you have to rewrite rules for each domain.

That this parsing approach works has been validated in practice.

CIRCE, Cont'd

The models produced by CIRCE include:

- **Entity models, almost UML**
- **Functional models, including SCR tables [Gervasi2000b]**
- **Behavior & time models**
- **Document models, i.e., grammatical structure**
- **Measurements of documents, system models, and processes**
- **Multiple views, involving other models [Ambriola2000]**

A less formal approach is taken by others who offer writing guidelines.

- **Guidelines for structuring a requirements specification [Kovitz1998, Fairley1985]**
- **Guidelines for writing more clearly [Dupré1998, Knuth1989]**
- **Guidelines for avoiding ambiguity [Berry2001]**
- **Guidelines for avoiding dangerous linguistic constructs [Rupp1997, Berry2000]**

[Abbott1983]
R.J. Abbott, “Program Design by Informal English Descriptions,” *CACM* **26**(11) (November 1983).

[Aguilera1990]
C. Aguilera and D.M. Berry, “The Use of a Repeated Phrase Finder in Requirements Extraction,” *Journal of Systems and Software* **13**(9), p.209–230 (1990).

[Ambriola2000]
V. Ambriola and V. Gervasi, “Supporting Multiple Views on Requirements,” pp. 321–330 in *Proceedings of the 6th Maghrebian Conference on Computer Sciences*, Fes, Morocco (November 2000).

[Anquetil1998]
N. Anquetil and T. Lethbridge, “Extracting Concepts from File Names; a New File Clustering Criterion,” pp. 84–93 in *Proceedings of the Twentieth International Conference on Software Engineering (ICSE’98)*, Kyoto, Japan (1998).

[Basili1997]
V.R. Basili, “Evolving and Packaging Reading Technologies,” *Journal of Systems and Software* **38**(1), p.3–12 (1997).

[Berry1987]
D.M. Berry, N.M. Yavne, and M. Yavne, “Application of Program Design Language Tools to Abbott’s Method of Program Design by Informal Natural Language Descriptions,” *Journal of Software and Systems* **7**, p.221–247 (1987).

[Berry2000]
D.M. Berry and E. Kamsties, “The Dangerous ‘All’ in Specifications,” pp. 191–194 in *Proceedings of 10th International Workshop on Software Specification & Design, IWSSD-10*, IEEE CS Press (2000).

[Berry2001]
D.M. Berry, E. Kamsties, D.G. Kay, and M.M. Krieger, “From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity,” Technical Report, University of Waterloo, Waterloo, ON, Canada (2001).

[Booch1991]
G. Booch, *Object Oriented Design*, Benjamin/Cummings, Redwood City, CA (1991).

[Bowden1998]
P.R. Bowden, L. Evett, and P. Halstead, “Automatic Acronym Acquisition in a Knowledge Extraction Program,” in *COMPUTERM Workshop at COLING-ACL’98*, Montréal, QE, Canada (1998).

[Comer1983]
J.R. Comer, “An Experimental Natural-Language Processor for Generating Data Type Specifications,” *SIGPLAN Notices* **18**(12), p.25–33 (December 1983).

[Denger2001]
C. Denger, J. Dörr, and E. Kamsties, “A Survey on Approaches for Writing Precise Natural Language Requirements,” IESE-Report, Fraunhofer IESE (2001).

[Dupré1998]
L. Dupré, *Bugs in Writing: A Guide to Debugging Your Prose*, Addison-Wesley, Reading, MA (1998).

[Enomoto1984a]
H. Enomoto, N. Yonezaki, M. Saeki, and H. Armata, "Formal Specification and Verification for Concurrent Systems by TELL," pp. 732–745 in *Advances in Artificial Intelligence, ECAI'84*, ed. T. O'Shea, Elsevier North-Holland (1984).

[Enomoto1984b]
H. Enomoto, N. Yonezaki, M. Saeki, K. Chiba, T. Takizuka, and T. Yokoi, "Natural Language Based Software Development System TELL," pp. 721–731 in *Advances in Artificial Intelligence, ECAI'84*, ed. T. O'Shea, Elsevier North-Holland (1984).

[Fairley1985]
R.E. Fairley, *Software Engineering Concepts*, McGraw-Hill, New York, NY (1985).

[Frakes1992]
W.B. Frakes and R. Baeza-Yates, *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall, Englewood Cliffs, NJ (1992).

[Fuchs1999]
N.E. Fuchs, U. Schwertel, and R. Schwitter, "Attempto Controlled English (ACE) Language Manual Version 3.0," Technical Report Nr. 99.03, Institut für Informatik der Universit~~ a~~t Zürich, Zürich, Switzerland (1999).

[Gervasi2000a]
V. Gervasi, "Environment Support for Requirements Writing and Analysis," Ph.D. Dissertation, TD-3/00, Dipartamento di Informatica, Università di Pisa, Pisa, Italy (2000).

[Gervasi2000b]
V. Gervasi and B. Nuseibeh, "Lightweight Validation of Natural Language Requirements," in *Proceedings of 4th IEEE International Conference on Requirements Engineering (ICRE'2000),*, Schaumburg, IL (19–23 June 2000).

[Goldin1997]
L. Goldin and D.M. Berry, "AbstFinder: A Prototype Abstraction Finder for Natural Language Text for Use in Requirements Elicitation," *Automated Software Engineering* **4**, p.375–412 (1997).

[Ishihara1993]
Y. Ishihara, H. Seki, and T. Kasami, "A Translation Method from Natural Language Specifications into Formal Specifications Using Contextual Dependencies," pp. 232–239 in *Proceedings of the First IEEE International Symposium on Requirements Engineering*, San Diego, CA (January 1993).

[Jackson1995]
M.A. Jackson, "Problems and Requirements," pp. 2–8 in *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, York, UK (March 1995).

[Kamsties2001a]
E. Kamsties, D.M. Berry, and B. Paech, "Living with Ambiguity in Industrial Requirements Specifications," Technical Report, Computer Science, University of Waterloo (2001).

[Kamsties2001b]
E. Kamsties, “Surfacing Ambiguity in Natural Language Requirements,” Ph.D. Dissertation, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany (2001).

[Knuth1989]
D.E. Knuth, T.L. Larrabee, and P.M. Roberts, *Mathematical Writing*, Mathematical Association of America, Washington, D.C. (1989).

[Kovitz1998]
B.L. Kovitz, *Practical Software Requirements: A Manual of Content and Style*, Manning, Greenwich, CT (1998).

[Laitinen1992]
K. Laitinen and T. Mukari, “DNN-Disciplined Natural Naming, A Method for Systematic Name Creation in Software Development,” pp. 91–100 in *Proceedings of 25th Hawaii International Conference on System Sciences, Vol. II: Software Technology*, IEEE Computer Society Press, Los Alamitos, CA (1992).

[Lecœuche2000]
R. Lecœuche, “Finding Comparatively Important Concepts Between Texts,” in *IEEE International Conference on Automated Software Engineering (ASE)*, Grenoble, France (11–15 September 2000).

[Leite1993]
J.C.S.P. Leite and A.P.M. Franco, “A Strategy for Conceptual Model Acquisition,” pp. 243–246 in *Proceedings of the First IEEE International Symposium on Requirements Engineering*, San Diego, CA (January 1993).

[Maarek1989]
Y.S. Maarek and D.M. Berry, “The Use of Lexical Affinities in Requirements Extraction,” in *Proceedings of the Fifth International Workshop on Software Specification and Design*, Pittsburgh, PA (May 1989).

[Rolland1992]
C. Rolland and C. Proix, “A Natural Language Approach for Requirements Engineering,” pp. 2577 in *Proceedings of Conference on Advanced Information Systems Engineering, CAiSE 1992*, Manchester, UK (12–15 May 1992).

[Rupp1997]
C. Rupp and R. Götz, “Sprachliche Methoden des Requirements Engineering (NLP),” in *PROFT # CONQUEST-1, First Conference on Quality Engineering in Software Technology*, Nürnberg, Germany (25–26 September 1997).

[Ryan1993]
K. Ryan, “The Role of Natural Language in Requirements Engineering,” pp. 240–242 in *Proceedings of the First IEEE International Symposium on Requirements Engineering*, San Diego, CA (January 1993).

[Saeki1987]
M. Saeki, H. Horai, K. Toyama, N. Uematsu, and H. Enomoto, “Specification Framework Based on Natural Language,” pp. 87–94 in *Proceedings of the Fourth International Workshop on Software Specification and Design*, Monterey, CA (April 1987).

[Salton1989]
G. Salton, *Automatic Text Processing: The Translation, Analysis, and Retrieval of Information by Computer*, Addison Wesley, Reading, MA (1989).

[Somé1996]
S. Somé, R. Dssouli, and J. Vaucher, "Toward an Automation of Requirements Engineering Using Scenarios," *Journal of Computing and Information* **2**(1), p.1110–1132 (1996).

[Srinivasan1992]
R. Srinivasan, "Thesaurus Construction," pp. 161–218 in *Information Retrieval: Data Structures and Algorithms*, ed. W.B. Frakes and R. Baeza-Yates, Prentice-Hall, Englewood Cliffs, NJ (1992).