Ambiguity defects as variation points in requirements

Laura Semini

joint work with Alessandro Fantechi and Stefania Gnesi

ISTI–CNR, Pisa, Italy
Universita’ di Pisa, Italy
Universita’ di Firenze, Italy
1. Ambiguities and their detection
   - Processing of Natural Language Requirements
   - Typical ambiguity defects
   - QuARS
2. Running example
   - Requirements for coffee machines
3. Software Product Lines and Feature Models
4. Ambiguity $\rightarrow$ Variability in requirements
   - Variability due to Vagueness
   - Variability due to Optionality
   - Variability due to Multiplicity
5. Variability Elicitation Process
6. Conclusions & Future work
Several studies proposing Natural Language Processing techniques to increase the quality of natural language requirement documents can be found in the literature.

Many works are aimed at ambiguity detection and resolution (ambiguous requirement can easily produce erroneous implementations), and formalization of requirements.

Hello, I am C 3PO, human cyborg relations
**Vagueness:** the sentence contains items having a no uniquely quantifiable meaning.

adequate, bad, clear, close, easy, efficient, far, fast, good, in front, near, recent, significant, slow, strong, suitable, useful, ...

**Optionality** the sentence contains an optional part (i.e. a part that can be considered or not)

possibly, eventually, in case, if possible, if appropriate, if needed, ...

**Multiplicity** The sentence has more than one main verb or more than one direct or indirect complement that describes the sentence.

and, and/or, or, ...
QuARS: Quality Analyser for Requirements Specifications

A tool for the analysis of natural language requirements developed at ISTI - CNR, Pisa

QuARS performs an analysis at sentence level, both syntactical and lexical, whose aim is to find the evidence of indicators of quality characteristics in the sentence, with reference to a given quality model.

These indicators are either lexical elements (verbs, adjectives) taken from user defined dictionaries or syntactical elements and constructs.

The dictionaries can be defined by the user.
Ambiguity detection in QuARS:

by lexical evaluation:
using lexical parsers to detect terms or wordings that are ambiguous (i.e. that may have multiple meanings according to the context).

Optionality, Vagueness, Subjectivity (similar, better, worse, having in mind, take into account, as... as possible), Weakness (can, could, may)

by syntactical evaluation:
using syntactical analysers to detect sentences having different interpretations on the basis of different derivation trees.

Implicitly (when the subject is generic rather than specific) and Multiplicity (if the sentence has more than one main verb or more than one direct or indirect complement that specifies its subject)
Running example: Requirements for coffee machines.

R1 After inserting a suitable coin, the user shall choose a beverage and whether to add sugar.

R2 Offered beverages are Coffee and/or Cappuccino and/or Tea.

R3 Coffee shall be offered by all products of the family.

R4 A ring tone shall be possibly rung after delivering a beverage.

R5 After the beverage is taken, the machine returns idle.
QuARS in action

The line number:
1. After inserting a suitable coin, the user shall choose a beverage and whether to add sugar.
   *is defective because it contains the wording: and*

The line number:
3. A ring tone shall be possibly rung after delivering a beverage.
   *is defective because it contains the wording: possibly*

The line number:
5. Offered beverages are coffee and/or cappuccino and/or tea.
   *is defective because it contains the wording: and/or*
(Software) Product Line Engineering
(Slides on SPL are partly based on slides by Maurice ter Beek, Klaus Pohl, Sven Apel and Slinger Jansen)

Develop a family of products (product line) using a shared platform or architecture (commonalities) and mass customization (variabilities)

Aim: maximize commonalities whilst minimizing cost of variations (i.e., of individual products), thus specifically facilitating (software) reuse in a predictive manner

Variability in terms of features:

- Features became popular in the 1960s in telecommunication with the advent of computer-controlled telephone switches; telecommunication software has been conceived in terms of features ever since.
- Currently, they are used to design SPL
Variation points and variants

A variation point represents an aspect of a product family that varies among the different products.

Example of variation point: the input modality of the user interface

A variant represents a specific configuration of a variable aspect that a product in a product family can have.

Variant 1: input via mouse and keyboard
Variant 2: input via touchscreen
Variant 3: input via t9 keyboard
Many (slightly) different definitions of feature have been provided in literature.

In our setting, we consider:

- End-user visible pieces of functionality that represent both
  - commonalities (e.g., mandatory, required) and
  - variabilities (e.g., optional, alternative)

Only specific combinations of features concern valid products
Features

What is a **feature**?
End-user visible behaviour or property of a system...
...that may be mandatory or optional and/or may have alternatives
Typically, only a subset of feature combinations is valid: a feature diagram describes how to select valid products.
Products and product lines

**Product**: valid feature combination (configuration)

**Product line**: set of valid feature combinations of a domain
Variability in requirements

R2 Offered beverages are Coffee and/or Cappuccino and/or Tea.
R3 Coffee shall be offered by all products of the family.

Can comprise the following four machines

1. A machine offering only Coffee
2. A machine offering Coffee and Cappuccino
3. A machine offering Coffee and Tea
4. A machine offering Coffee, Cappuccino and Tea
Variability due to **Vagueness**

After inserting a **suitable** coin, the user shall choose a beverage and whether to add sugar.
Variability due to Optionality

A ring tone shall be possibly rung after delivering a beverage.
Variability due to Multiplicity

Multiplicity may come in different flavors:

- exclusive or
  the alternatives are mutually exclusive.
- weak or
  all the alternatives are optional.
- or
  at least one of the alternatives should be present, no matter which.
- and/or
  at least one of the alternatives should be present, it matters which.
- and
  all conjuncts must be present.
Variability due to Multiplicity: case study

[R2] Offered beverages are Coffee and/or Cappuccino and/or Tea.

[R3] Coffee shall be offered by all products of the family.

[R1] After inserting a suitable coin, the user shall choose a beverage and whether to add sugar.
Variability Elicitation Process

A process to go:

From requirements ... ... to Feature Models

[R1] After inserting ...
[R2] Offered beverages ...
[R3] Coffee shall ...
[R4] A ring tone ...
[R5] After the beverage ...
Steps of the process

1. Run a tool for ambiguity detection in NL requirements.
2. Expert judgment to:
   1. Remove false positives.
   2. Substitute actually ambiguous requirements by more precise ones (and cycle back to analysis).
   3. Partition the remaining ones, which are candidate variability indicators, wrt different defect types.
3. Build a feature model capturing the identified variation points.
Process as an Activity Diagram

The process can be modelled by the flow:

- Change requirements
- Run an ambiguity detection tool
  - [there are actual ambiguities]
  - [there are false positives]
  - [else]
    - Remove false positives from the blacklist
- [else]
- [else]
- Specify range
  - [configuration parameter]
  - [there is a(ano)ther vague requirement]
  - [variability]
- Specify values and add FM fragment
- [and/or]
- [and]
- [or]
- [there is a(ano)ther multiplicity]
- [else]
- [else]
- [else]
- [else]
- [and/or]
- add FM fragment
  - [and]
- add FM fragment
- add FM fragment
- add FM fragment
- add FM fragment
- [mutual exclusion]
- [relative to another feature F]
- [absolute]
- [there is a(ano)ther optionality]
- add FM fragment to family root
- add FM fragment to F
- [else]
- edit the FM if needed
The process can be automated by a tool. In particular the third step could be supported by an interactive tool building FM fragments:

**For vague requirements:** the noun over which the vagueness-revealing word predicates, coin in our case, as root and the determined values as sub-features.

**For multiplicity,** the tool can offer a default choice, e.g. subject of the sentence, as root.

**For optionality,** the root of the fragment may be the family root or a feature in the model (when the optional feature is a sub-feature)
Conclusions & Future work

Conclusions

- Approach to extract variability issues from a requirements document.
- Ambiguity in requirements can be caused by the need to postpone choices for later decisions in the implementation of the system.
- Ambiguity becomes a means to enlighten possible variation points in an early phase of software and system development.

Current and Future work

- Implement the tool taking the output of QuARS and building the feature model.
- Evaluate approach and tool on real world examples (we are very interested in receiving case studies)