Applying the QuARS tool to detect variability

Alessandro Fantechi
Dipartimento di Ingegneria dell'Informazione,
Università di Firenze
Firenze, Italy
alessandro.fantechi@unifi.it

Stefania Gnesi
Istituto di Scienza e Tecnologie dell'Informazione "A.Faedo",
Consiglio Nazionale delle Ricerche,
ISTI-CNR
Pisa, Italy
stefania.gnesi@isti.cnr.it

Laura Semini
Dipartimento di Informatica,
Università di Pisa
Pisa, Italy
semini@di.unipi.it

ABSTRACT

In this demo paper we present how to use the QuARS tool to extract variability information from requirements documents. The main functionality of QuARS is to detect ambiguity by means of a Natural Language (NL) analysis in requirements. In fact, ambiguity defects that are found in a requirement document may be due to intentional or unintentional references to issues that may be solved in different ways, so making possible to envision a family of different products starting from an ambiguous requirement document.

ACM Reference format:
Alessandro Fantechi, Stefania Gnesi, and Laura Semini. 2019. Applying the QuARS tool to detect variability. In *Proceedings of 23rd International Systems and Software Product Line Conference, Paris, France, 9–13 September, 2019 (SPLC'19),* 5 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

QuARS is a tool able to perform an analysis of Natural Language (NL) requirements in a systematic and automatic way by means of natural language processing techniques: the focus is on the detection of ambiguity defects, according to the process depicted in Figure 1, by identifying candidate defective words stored in dictionaries.

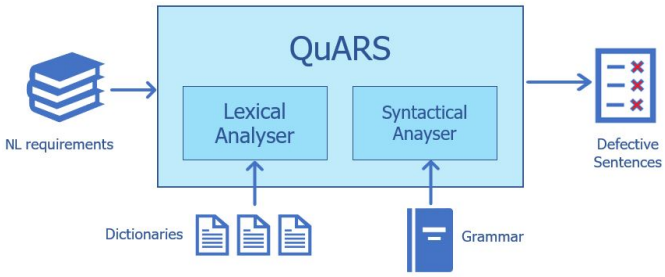


Figure 1: QuARS Process

SPLC'19, 9–13 September, 2019, Paris, France

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In [1–3] we have shown that ambiguity defects in requirements can in some cases give an indication of possible variability, either in design or in implementation choices or configurability aspects. In fact the ambiguity defects that are found in a requirement document may be due to intentional or unintentional references made in the requirements to issues that may be solved in different ways: the different implementations may give rise to a family of different products, instead of just a single product. In [3] we proposed a first classification of the forms of linguistic defects that indicate variation points, and we described a possible mapping from ambiguity or under-specification defects to fragments of feature models.

We therefore have used the analysis ability of QuARS to elicit the potential variability hidden in a requirement document. The process followed by QuARS for detecting potential variabilities is described below:

- A NL requirement document is given in input to QuARS to be analyzed according to its lexical and syntactical analysis engines, in order for identifying ambiguities.
- The detected ambiguities are analyzed in order to distinguish among false positives, real ambiguities, and variation points.

In this paper we illustrate as a demo how the above process is applied to detect variability in NL requirements using a simple case study.

2 RUNNING EXAMPLE

The case study used as running example is a (simplified) e-shop, for which we consider the following requirements:

R1 The system shall enable a user to enter the search text on the screen.
R2 The system shall display all the matching products based on the search.
R3 The system possibly notifies with a pop-up the user when no matching product is found on the search.
R4 The system shall allow a user to create his profile and set his credentials.
R5 The system shall authenticate user credentials to enter the profile.
R6 The system shall display the list of active and/or the list of completed orders in the customer profile.
R7 The system shall maintain customer email information as a required part of customer profile.
R8 The system shall send an order confirmation to the user through email.
R9 The system shall allow a user to add and remove products in the shopping cart.

R10 The system shall display various shipping methods.
R11 The order shall be shipped to the client address or, if the "collect in-store" service is available, to an associated store.
R12 The system shall enable the user to select the shipping method.
R13 The system may display the current tracking information about the order.
R14 The system shall display the available payment methods.
R15 The system shall allow the user to select the payment method for order.
R16 After delivery, the system may enable the users to enter their reviews or ratings.
R17 In order to publish the feedback on the purchases, the system needs to collect both reviews and ratings.
R18 The "collect in-store" service excludes the tracking information service.

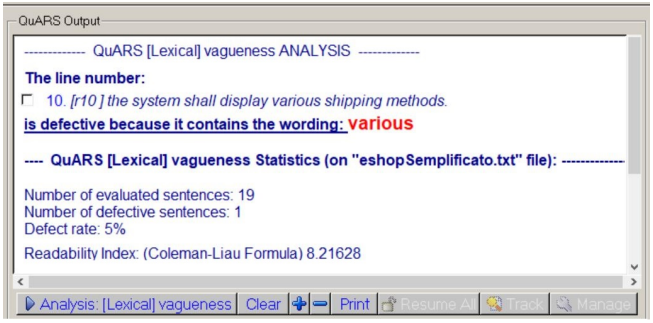
3 USING QUARS TO EXTRACT VARIABILITY

Looking at the set of requirements given in Sec.2 we can notice that there are a number of ambiguity defects, that are related to some words (the underlined ones) that introduce a possibility of different interpretation. In the remaining part of the section we will show the results of applying QuArs to this set of requirements to automatically detect ambiguity, and how these defects relate to variability and can be used to indicate variation points.

3.1 Vagueness

Vagueness means that the requirement contains words having no uniquely quantifiable meaning and therefore admits borderline cases, e.g., cases in which the truth value of the sentence cannot be decided. As an example: *the C code shall be* **clearly** *commented.* The QuARS Vagueness dictionary contains words like: *adequate, bad, clear, close, easy, far, fast, good, in front, near, recent, various, significant, slow, strong, suitable, useful, ...*

Running QuARS we find one vague requirement:



"Various" is a vague term, here indicating a variability about the different shipping methods that can be implemented.

In general, a vague word abstracts from a set of instances, that are considered as the "various" ones, and the process of requirement refinement will make these instances explicit. In terms of features, vagueness results in the introduction of an abstract feature (Fig. 2(a)). Once instances will be made explicit, they will be represented by sub-features, one for each instance.

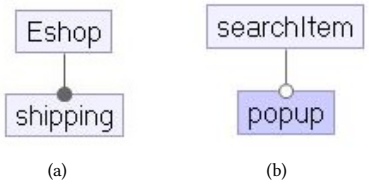
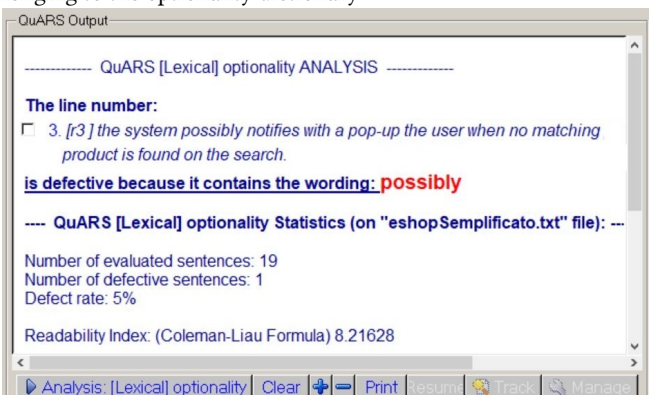


Figure 2: (a)-(b) Feature diagrams for vagueness and optionality

3.2 Optionality

Optionality occurs when a requirement contains an optional part, i.e. a part that can be considered or not: *the system shall be..,* **possibly** *without...* Example of Optionality-revealing words are: *possibly, eventually, in case, if possible, if appropriate, if needed, ...*

In our e-shop we find one requirements containing a term belonging to the optionality dictionary:



Optionality of a requirement is naturally expressed in a feature diagram with an optional feature. In the example, it has been recognized that the pop-up notification is an optional feature, as expressed by the fragment shown in Fig. 2(b).

3.3 Multiplicity

Lexical multiplicity means that the requirements does not refer to a single object, but addresses a list of objects, typically using disjunctions or conjunctions (*or, and, and/or*), like in .. *opens doors* **and** *windows...*

In particular, a requirement with an **or** is not precise, since it leaves several possibilities open, and hence different products compliant to such requirement:

- **implicitly exclusive or:** the alternatives are mutually exclusive. In this case, the corresponding features are declared as "alternative", with a partial diagram as the one used for vagueness.
- **weak or:** all the alternatives are optional.
- **logical** $\vee$**:** at least one of the alternatives should be present, but it is irrelevant which one.

Construct **and/or** is basically equivalent to the third interpretation of or, i.e. it requires that at least one of the alternatives should

be present. The manual analysis can refine the interpretation to say which one of the alternatives is mandatory, and the other optional.

An **and**, although recognized as a multiplicity, simply shows that all alternatives are mandatory: hence it is not really a variability indication, although it can be modelled with a feature diagram as well.

Requirement **R4**, matches exactly the case described above, conjunction of customer profile and credentials is represented by the fragment in Fig. 3(c). On the contrary, **R9** is a false positive. Including **and** in the multiplicity dictionary or not is a debatable issue: on the one side, it defines possible fragments of the feature diagram –a feature with two or more mandatory subfeatures–, on the other side, the analysis with QuARS may return many false positives. However, since adding or removing a word from a dictionary takes few seconds, the analyst can switch from one solution to the other.

The disjunction "rating or review" in **R16** is represented by the fragment in Fig. 3(b) in this case we used a **weak or** because of the weakness "may" (see section 3.4).

In the case of **R6**, the analyst has resolved the variability telling that the list of active orders should be mandatory, while the list of completed orders can be considered optional (see Fig. 3(a)).

Finally, **R11** defines the instances, "homeAddress" and "store" needed to make concrete the abstract feature "shipping" in Sect. 3.1: Their variability information is discussed in section 3.5. Requirement **R17** is dealt with in section 3.6.

3.4 Weakness

Weakness occurs when the sentence contains a "weak" verb. A verb that makes the sentence not imperative is considered weak (i.e. *can, could, may, ..*). For instance "the initialization checks ***may be*** reported . . . " is a weak sentence.

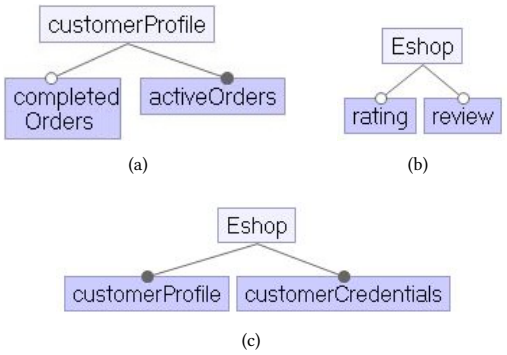
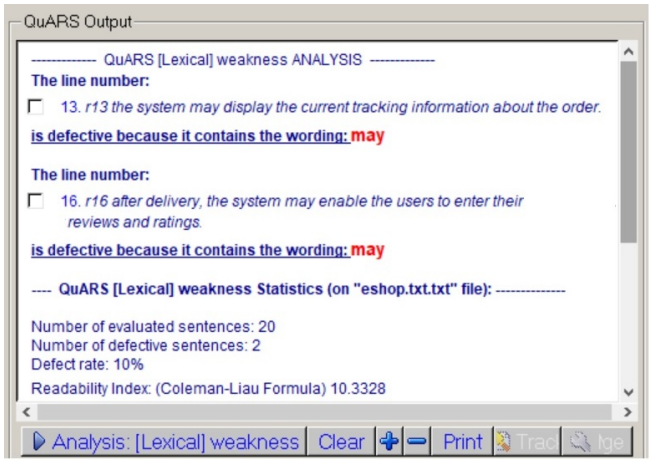


Figure 3: (a)-(c) Feature diagrams for multiplicity



These two defective sentences clearly introduce an optionality, represented in Fig. 4(a) and 3(b), respectively.

(a)

Figure 4: (a) Feature diagram for weakness

3.5 Variability

In [2], we also exploited the capability of QuARS to add dictionaries for new indicators, namely variability related terms (*if, where, whether, when, choose, choice, implement, provide, available, feature, range, select, configurable, configurate*) and constraints identifiers (see Sect. 3.6).

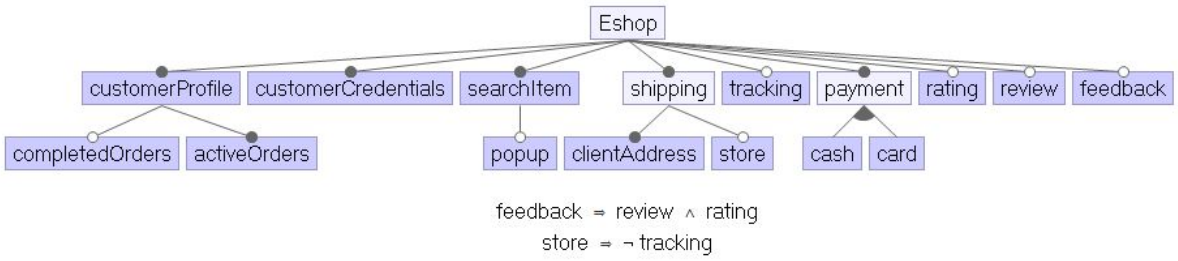
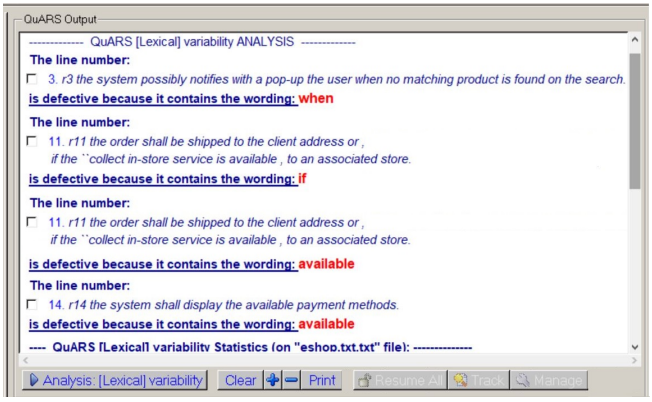
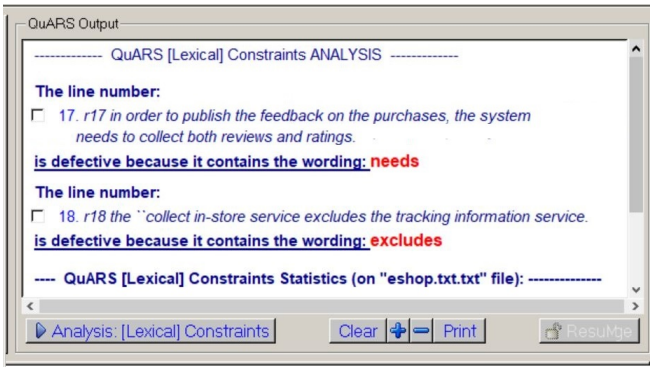


Figure 5: Feature model for eshop





R3 is a false positive. **R11** and **R14** are actual variabilities. **R11** says that "ship to store" depends on the optional service "collect-to-store". Feature shipping is mandatory: at least a concrete subfeature must be present in each product. The choice, in this case, is to define "clientAddress" feature as mandatory, and "ship to store" optional, as expressed in the fragment of Fig. 6(a).

R14 is recognized to be a variability, that can be expressed by an *or* of the different payment methods that can be adopted for the system (see Fig.6(b)). Different payment methods were not specified in the original requirements, and have been expanded in this example as payment by card or payment by cash.

Notice that in both cases above the word "available" indicates a variability, but the nature of the variability is different because in the first case it is found inside an "if" context.

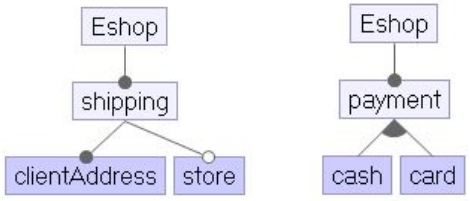


Figure 6: (a)-(b) Feature diagrams for variability indicators

3.6 Constraints

Additional *cross-tree constraints* that is **requires**, indicating that the presence of one feature implies the presence of the other, and **excludes**, indicating that no system may contain the two features at the same time, may be detected looking at the occurrence of constraint-revealing terms such as: expect, need, request, require, excludes ….

In **R17** the term "needs" appears to indicate that a "publishFeedback" is an optional feature and that a *requires* cross-constraint has to be included in the model:

$$feedback \Rightarrow review \wedge ratings$$

Requirement **R18** adds another cross-constraint to the model due to term "exclutes":

$$store \Rightarrow \neg tracking$$

Gluing together all the fragments extracted so far, we have built the feature diagram in Figure 5.

REFERENCES

[1] A. Fantechi, A. Ferrari, S. Gnesi, and L. Semini. Hacking an ambiguity detection tool to extract variation points: an experience report. In R. Capilla, M. Lochau, and L. Fuentes, editors, *Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2018, Madrid, Spain, February 7-9, 2018*, pages 43–50. ACM, 2018.

[2] A. Fantechi, A. Ferrari, S. Gnesi, and L. Semini. Requirement engineering of software product lines: Extracting variability using NLP. In G. Ruhe, W. Maalej, and D. Amyot, editors, *26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018*, pages 418–423. IEEE Computer Society, 2018.

[3] A. Fantechi, S. Gnesi, and L. Semini. Ambiguity defects as variation points in requirements. In *Proc. of the Eleventh International Workshop on Variability Modelling of Software-intensive Systems*, VAMOS '17, pages 13–19, New York, NY, USA, 2017. ACM.

A APPENDIX PART 1: HOW THE TOOL WILL BE PRESENTED AS PART OF A MAIN CONFERENCE SESSION

During the main conference session we will use slides, with the following outline:

(1) we first present the tool, QuARS;
(2) we motivate the use of QuARS for eliciting variation points;
(3) we present the e-shop use case;
(4) we use the screenshots used in this paper to illustrate how QuARS is used in practice.

B APPENDIX PART 2: HOW THE TOOL WILL BE PRESENTED AT A DEMO BOOTH

During the demo we will run QuARS on the running example or on any requirement document provided by the conference participants. Documents must be:

- written in english;
- in .txt format.

Moreover, we can use other requirement documents, we have experience with:

European Integrated Railway Radio Enhanced Network The "Eirene" document has been issued by the GSM-R Functional Group and it specifies the functional requirements for a digital radio standard for the European railways. The document includes 475 requirements.

Library The second document, "Library", was prepared by the Galecia Group, a company specialised in libraries and organizations supporting libraries. It describes the functional and nonfunctional requirements for the System Administration Module of the Integrated Library System of a urban library system. It includes 94 requirements.

Blit which is a draft of the functional specification of the requirements of a business project management tool, required by a company for re-writing its core Laboratory Information System to improve the performance. It includes 55 requirements.

ERTMS: train control system This document defines the functional requirements for ERTMS/ETCS (European Rail Traffic Management System / European Train Control System), issued by the European Railway Agency in June 2007. The document includes 96 requirements of a control system that supports the driver of a train: it provides the driver with information needed for the safe driving of the train, and it is able to supervise train and shunting movements.

People by Temperament: social web application This document comes from Plat_Forms, an international academic-industrial programming contest. It specifies a community portal where members can find others with whom they might like to get in contact: people register to become members, take a personality test, and then search for others based on criteria such as personality types, likes/dislikes etc. The documents includes 325 requirements.

DigitalHome: home automation system This requirements document specifies a *Smart House*, called DigitalHome (DH). The case study, including 151 requirements, has been developed and used by a team of 5 students of a computing curriculum, as part of a US National Science Foundation grant. The DH system allows a home resident to manage devices that control the environment of a home. The user communicates through a web page on a web server. The DH web server communicates, through a wireless gateway device, with the sensor and controller devices in the home.

C VIDEO

The video demonstrating the core features of the tool is at:
https://fmt.isti.cnr.it/video.avi