Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Viewpoint Resolution: A Critical Evaluation

Mohammed Messaoudi
IMSIU, College of Sciences, Riyadh, Saudi Arabia
Corresponding Author: Mohammed Messaoudi, **E-mail**: mmessaoudi@imamu.edu.sa

| ABSTRACT
Viewpoints-based requirements engineering is an active area of research. This paper provides a critical evaluation of a particular technique for the early validation of requirements using viewpoints. The technique is limited to the syntactic analysis of requirements and lacks a conflict resolution strategy. This paper describes an approach to the very early validation of requirements based on learning about the viewpoints and building models of their behaviour. The method is a collection of domain-independent heuristics to build internal models of the viewpoints that record their performance in providing information, assessing information, and resolving conflicts between viewpoints.

| KEYWORDS
Requirements Engineering, Viewpoint-based Requirements Engineering, Early Validation, Viewpoint Analysis, Viewpoint Resolution.

| ARTICLE DOI: 10.32996/jcsts.2022.4.1.5

1. Introduction

Viewpoints-based requirements engineering is an active area of research. A viewpoint-based method is seen here as a process of identifying viewpoints, reasoning within a viewpoint, reasoning between different viewpoints, and revising a viewpoint [Messaoudi, 2013]. Leite, J. [Leite 1988] proposed Viewpoint Analysis as a method for early requirements validation. He contends that by describing the same domain, from two different perspectives, in the same language using the same vocabulary and then examining the differences between the resulting descriptions, the chances of detecting problems are 'enhanced'. Leite states that his method provides a better approximation between the universe of discourse and the gathered facts than the existing methods. This chapter evaluates Leite's method from the early validation perspective and from the perspective of multiple viewpoints usage.

Brackett [Brackett, 1990] describes Leite's thesis as 'valuable to anyone working seriously in developing requirements definition methods'. However, Leite's actual contribution needs careful examination from the point of view of requirements validation. Figure 1 represents a SADT (Structured Analysis and Design Technique [Ross, and Schoman,1977] model of Leite's viewpoint resolution method. Leite's main work, however, is concentrated on the first part of the method, viewpoint analysis, to support fact-validation. The second part, viewpoint reconciliation, to support communication, is deferred as a difficult problem.

2. Viewpoint Analysis

Viewpoint analysis consists of:

a) View construction with the following guidelines:

- find the facts
- express the facts using the keywords of the application domain
- classify the facts into object facts, actions facts, and agent facts

• codify the facts using the VWPL language.

b) Static analysis:

• Compare (syntactically) the different descriptions (from the same viewpoint) then compare the different descriptions from different viewpoints.

• Classify the differences into missing information and wrong information as well as inconsistencies.

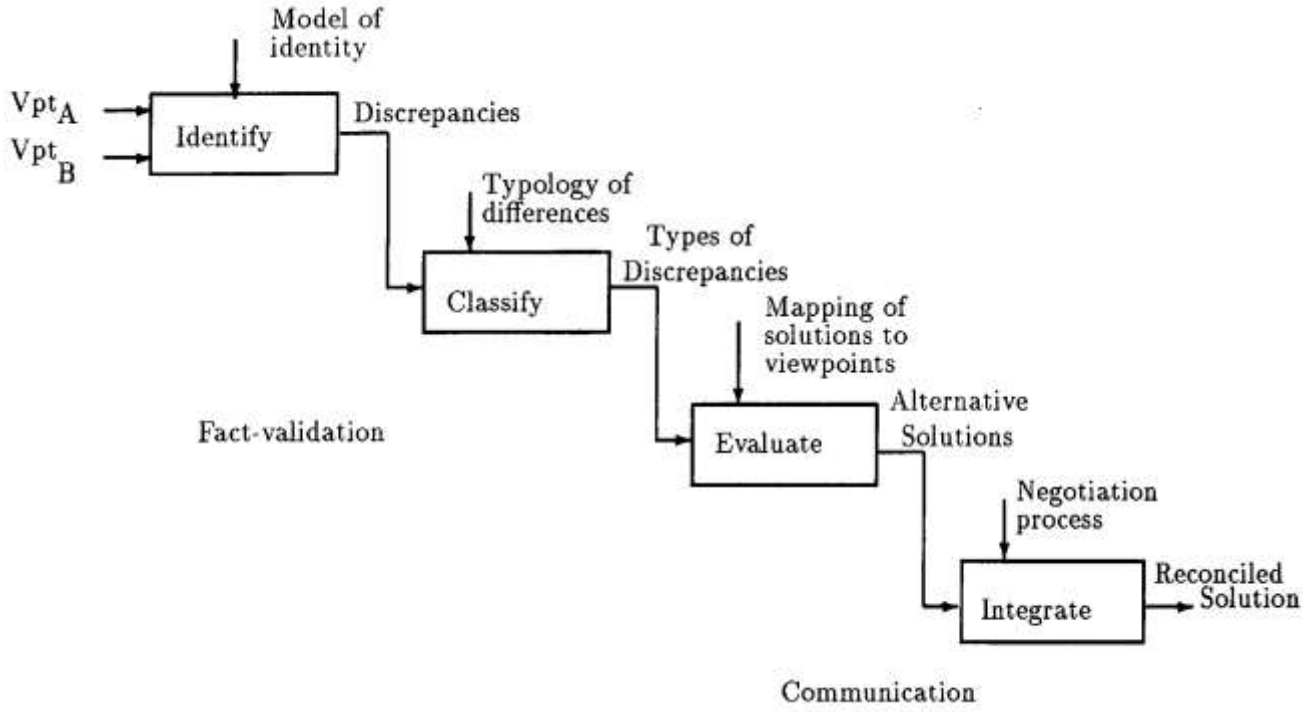

Figure 1: Viewpoint Resolution

2.1 View Construction

In order to construct a view, an analyst (viewpoint) finds the facts using his/her favourite methods, representations and tools then describe the problem using three perspectives and two hierarchies. The perspectives are the actor perspective, the data perspective, and the process perspective. An example of perspective modelling is CORE's 'Data Structuring' and 'Action Structuring' used to model viewpoints [Mullery,1977,1985] and SADT's datagrams and actigrams. The idea behind actor modelling is to model using the perspectives of those who are responsible for the processes, i.e., human agents or devices. The hierarchies are the is-a hierarchy and the parts-of hierarchy. A view is the integration of perspectives and hierarchies. The construction of a perspective is based on the assumption that the analysts use the application vocabulary (keywords of the application domain) when describing their views.

A perspective is expressed in the VieWPoint Language (VVVPL). VWPL is a rule-based language with a predefined structure for the construction of rules. A rule is made of facts (a fact is a relationship between keywords of the application domain). A fact is composed of a fact-keyword and a fact-attribute. For example, the fact (book =book-id author = title) has book as the fact-keyword and has as attributes book-id, author, and title. Facts are classified according to type and class. There are three types of facts: the object fact, the action fact, and the agent fact. Classes are the different roles each fact may play in a rule. For example, the action "check-out a copy of a book" from a library problem can be represented by the following rule:

(21 ((check-out =borrower =copy)

(book =author = title =copy)

(library-copy =copy)

⟶

(<not> (copy-borrower =borrower =copy)))

(($delete-from wm (check-out =borrower =copy))

($add-to wm (copy-borrower =borrower =copy))))

(is-a (1 (user borrower library – staff)))

(parts-of (2 (book author title copy)))

2.2 Static Analysis

Static analysis is the syntactic comparison of different perspectives and different views using a pattern and partial matching. Static analysis has two tasks: finding which rules are similar and, once rules are paired, identifying and classifying the discrepancies between them (see Figure 2). The discrepancies are classified as follows:

• Incorrectness: contradiction between facts of the different rule sets.

• Incompleteness:
  - missing rules:
  - missing facts
  - incomplete hierarchies with respect to rule facts

• Inconsistency
  - contradiction between a fact and the hierarchy
  - redundancy in the same ruleset.

Static analysis is driven by a set of heuristics built into a support tool called a static analyzer. There are four types of heuristics used in the static analyzer (see Figure 2). The partial matching heuristics are used in finding out similarities between facts from the different rule sets. The scoring heuristics represent a scoring scheme for compounding different matching scores to find out the possible rule pairs. The scoring evaluation heuristics are used to identify the best pairings between rules. The classification heuristics use the hierarchies to define the types of the critics produced: missing information, wrong information, or inconsistencies. An exemple of a heuristics [Leite, 1989,1991]:

If a fact is rule x from viewpoint_A is a leaf in the parts-of hierarchy, AND a fact in rule y from viewpoint_B is a leaf in a parts-of hierarchy, AND the hierarchy root is the same (leaves of the same hierarch).

→

the facts are in contradiction (one of the rules has wrong information).

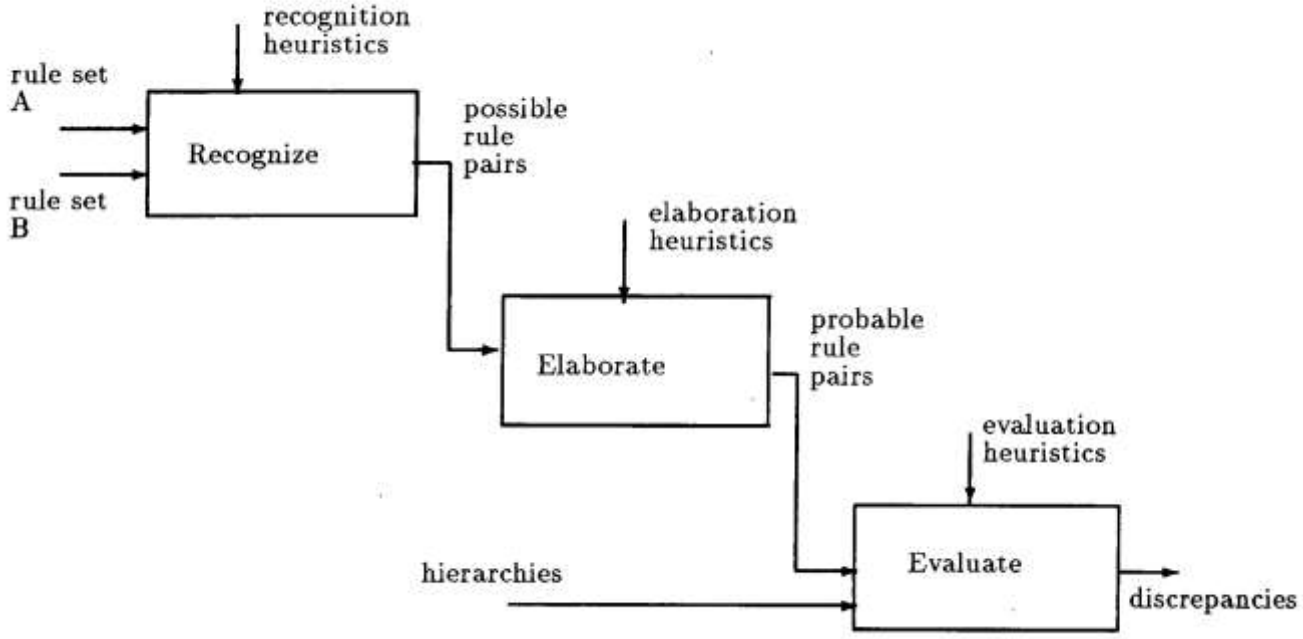

Figure 2: The static analyzer heuristics

3. The Limitations of the Method

The shortcomings of the viewpoint analysis method can be summarized in the following points:

1. The previous chapter concludes with the observation that under the severe restrictions imposed by Leite in his method (viewpoints should consider the same topic, use a common vocabulary, and use the same language, VWPL which imposes constrains how the rules should be expressed) the viewpoint analysis is almost reduced to a single perspective method in which different viewpoints must observe a domain almost from the same angle. But in evaluating his method, Leite acknowledges that the assumptions need to be relaxed. He suggests that a negotiation process between the viewpoint holders should occur before applying his method, i.e., that the facts should be analyzed before their codification into VWPL. In other words, a sort of viewpoint resolution process should take place before the application of his method.

2. The method is syntax-oriented. Using the method, similar concepts with different syntactic forms can be declared different or even contradictory, and different concepts with similar syntactic forms can be found to be the same.

3. The method cannot say anything about two statements, considered to be consistent but which are not correct with respect to the universe of discourse

4. The method does not have a resolution/reconciliation strategy, and it is difficult to 'retrofit' a resolution strategy in the technique's present form (one may draw an analogy with specification reuse [Finkelstein,1988]). The main reasons for the difficulties are:

(a) The absence of a strategy for identifying the viewpoints' holders (actors) necessary to identify the universe of discourse within which the facts are assessed.

(b) The absence of explicit links between the universe of discourse and the gathered facts (except the assumption that the application vocabulary is used in the process of codifying the facts), necessary for validation (links can be used to trace the roots of conflicts back in the universe of discourse [Easterbrook,1991, Robinson, 1989, Fickas and Nagarajan, 1988]).

(c) The non-involvement of the users directly in the validation process. The method is oriented to those who build the requirements model, who are usually the people on the supply side. This leads to ineffective communication. Communication plays an important role in conflicts resolution [Elam *et al., 1987*].

4. Case Studies

Since there are no restrictions on what representation should be used to record the facts before their codification into VWPL, we assume natural language is the 'natural' choice. Two counterexamples to Leite's strategy are chosen: the first represents two views with similar meanings but with slightly different syntax. The second example represents two views with different meanings but with similar syntax. The texts were obtained from two different portions of specifications produced by CORE and Remora, respectively [5]. The CORE enables the reader to trace the derivation of each dataflow-fragment from the text representing the needs statement because each such fragment is indexed by line into the text. The CORE specification is then used to guide the derivation of the second text from the Remora specification.

The first example concerns a system for online monitoring of patients in an intensive care unit. The following are two possible views about the initial activation of the monitoring system.

View.One

A hospital staff member activates the monitoring system when a new patient is admitted. The monitoring system then prompts the staff to set up the initial patient's data.

View_Two

When a new patient is admitted, a hospital staff member is responsible for activating the monitoring system and initializing the patient's data.

The following are possible representations of the views in VWPL:

Viewpoint_One:

1 (activate-system =patient-admitted) (nurse =status)
(system = patient-monitoring-system)
.....▶

$delete-from-wm (activate-system =patient-admitted)
$add-to-wm (activate-request =initial-prompt)
: activate-system (of type action) is the pre-condition input
: nurse (agent) and system (object) are invariants
: activate-request (object) is the post-condition output
; patient-admitted, status, patient-monitoring-system,
: and initial-prompt are attributes.

2 (set-up-patient-data =patient-identity =initial-prompt)
(nurse =status)
(patient-admitted =patient-identity)
(system = patient-monitoring-system)
$delete-from-wm (set-up-patient-data =patient-identity=initial-prompt)
$add-to-wm (patient-data =initial-patient- data)

Hierarchies:

parts-of (patient-data patient-identity initial–patient-data)
is-a (staff nurse)
: patient-identity initial-patient-data are parts of patient-data
: staff is a generalization of nurse

Viewpoint.Two:

10 (activate-monitoring-system =new-patient-admission)
(medical-staff-member =name =rank)
(monitoring-system)
⟶
$delete-from-wm (activate-monitoring-system = new-patient-admission)

20 (initialize-data =new-patient-admission)
(medical-staff-member =name =rank)
⟶
$delete-from-wm (initialize-data =new-patient-admission)
$add-to-wm (patient-data =patient-id =initial-unsensed-parameters
=Upper-bound-monitored-params = etc)

Hierarchies:

is-a (staff medical-staff-member hospital-staff-member
parts-of (patient-data patient-id initial-unsensed-parameters Upper-bound-monitored-params:
Using Leite's static analysis, the possible rule pairs are (Rulel, Rule10) and (Rule2, Rule20). Comparing Rulel and Rule10 syntactically, we obtain the following discrepancies:

1. The facts (nurse =status) from Rulel and (medical-staff-member =name =rank) from Rulel0 are in contradiction because they are of the same type and of the same class (agent-invariant), syntactically different and have the same root in the is-a hierarchy.
2. The attributes of the system in Rulel and the monitoring-system in Rulel0 do not correspond while they syntactically match.
3. The facts (activate-system =patient-admitted) and (activate-monitoring-system = new-patient-admission) are a perfect match.
4. Rulel0 is missing the information (activate-request =initial-prompt)
5. The facts (nurse =status) from Rule2 and (medical-staff-member =name =rank) from. Rule20 are in contradiction.
6. Rule20 is missing the fact (patient-admitted =patient-identity)
7. The attributes of patient-data in rides Rule2 and Rulr20 do not correspond.

The second example concerns a system for monitoring a high-security intensive care unit. The monitoring system should poll sensors at regular intervals. When sensors detect undesirable events, the system notifies the emergency services of exceptional conditions. The following are two possible interpretations of the problem:

Text_Three

A system for monitoring patients polls sensors every 60 seconds. If the sensors detect abnormalities, the system notifies the emergency services of exceptional health conditions.

Text_Four

A system for monitoring alarms polls sensors every 60 seconds. When the sensors detect abnormalities, the system notifies the emergency services of exceptional security conditions.

Viewpoint_One

3 (notify =alarm-buzzer-sounding) (exceptional-condition =patient)
(patient-monitoring-system =monitoring-data)
(sensor =location =type =60-seconds)
⟶
$delete from wm (notify =alarm-buzzer-sounding)
$add to wm (alarm =alert-alarm-state =audi-visual-alarm)
(emergency-services =health}

Hierarchies:
(is - a (emergency-services doctor)
(parts-of (monitoring-data blood-pressure temperature))

Viewpoint_Two

4 (notify =alarm-sounding) (exceptional-condition =alarm)
(alarm-monitoring-system =monitoring-data)
(sensor =location =type =60-seconds)
⟶
$add to wm (alarm =audio-visual-alarm) (emergency-services =security)
$delete from wm (notify =alarm-sounding)

Hierarchies:
(is - a (1 (alarm-state alert-alarm-state safe-alarm-state)
(emergency services police firemen)
(parts-of (monitoring-data lights)

Similarly, applying static analysis heuristics to rules 3 and 4, the following observations can be made:

1. Rules 3 and 4 should, according to the method, be declared as 'missing rules' because they are referring to two different parts (patient monitoring and alarm monitoring) of the same system, but
2. The heuristics hinted at some 'discrepancies' between the two views. For example, the attributes of the facts (exceptional-condition =alarm) and (exceptional-condition =patient) do not correspond.
3. The fact 'the monitoring system polls sensors every 60 seconds' is consistent with the rest of the facts but is 'critically' wrong.

5. Summary

Leite uses the viewpoint approach to support an early requirements validation by pointing out discrepancies in a problem description. The viewpoint analysis method is evaluated from the early validation perspective and from the perspective of multiple viewpoints usage. As shown earlier, each limitation of the viewpoint analysis method has been reflected in the quality of requirements validation, i.e., the type and quality of the discrepancies; in understanding a problem, the method can point out and help resolve them. It has been shown, through two case studies, the limitations of syntactic analysis of information; the absence of a strategy for identifying viewpoints (actors) that make up the universe of discourse lead to the absence of a universe of discourse within which information is validated. This, in turn, leads to the inability of the method to detect incorrect information with respect

to the universe of discourse and to the absence of explicit links between the gathered information and the universe of discourse necessary for traceability. The absence of a conflict resolution strategy and the non-involvement of the users directly in the validation process does not help communication necessary for resolving open issues. The hypothesis of this paper is that by improving the viewpoint resolution process, as proposed by Leite, the validation process could be improved as a result. An alternative technique will be proposed in a future paper.

References

[1] Messaoudi, M (2013), Requirements Engineering Through Viewpoints, *International Journal of Computing Academic Research,2(2)*
[2] Leite, J., (1988), Viewpoints Resolution in Requirements Elicitation, PhD thesis, *Department of Computer Science, University of California,* Irvine.
[3] Brackett, J.W, (1990). Sofware Requirements, in Standards, Guidelines, and examples on System and Software Requirements Engineering, M. Dorfman and R. H. Thayer (ed.).
[4] Ross, D.T. and Schoman, K.R., (1977): Structured Analysis for Requirements Definition, *IEEE Trans. Software Engineering, SE-3, (1)*.
[5] Fickas, S. and Nagarajan, P, (1988). Critiquing Software Specification, *IEEE Software, 1988 37-46*.
[6] Finkelstein, A. and Potts, C. (1985). Evaluation of Existing Requirements Extraction Strategies, Alvey FOREST project,
[7] Leite, J., (1989). Viewpoint Analysis: A Case Study, *ACM SIGSOFT Engineering Notes, 14(3*) .
[8] Leite, J. and Freeman, P., (1991): Requirements Validation Through Viewpoint Resolution, *IEEE transactions on Software Engineering 17(12)*.
[9] Finkelstein, A., (1988). Reuse of Formatted Requirements Specifications, *Software Engineering Journal, 186-197*.
[10] Easterbrook, S., (1991). Elicitation of Requirements from Multiple Perspectives, *PhD thesis. Department of Computing, Imperial College of Science, Technology, & Medicine, University of London*.
[11] Mullery, G., (1979). CORE - A method for Controlled Requirements Expression, *Proc. of Fourth IEEE Int. Conf. on Soft. Eng., Munich, Germany*.
[12] Robinson, W.N., (1989). Integrating Multiple Specifications using Domain Goals, *ACM SIGSOFT Engineering Notes, 14(3) 219-226*.
[13] Mullery, P., (1985). Acquisition-Environment, in Distributed Systems: Methods and tools for the specification, *Springer-Verlag*.
[14] Elam, J.J., Diane B. Walz, D.B., Krasner, H., and Curtis, B., (1987). A Methodology for Studying Software Design Teams: An Investigation of Conflict Behaviors in the Requirements Definition Phase, *2nd Workshop on Empirical Studies on Programmers, 83-99*