An Application of Paraconsistent Annotated Logic for Design Software Testing Strategies

Marcos Ribeiro do Nascimento¹, Luiz Alberto Vieira Dias¹, João Inácio Da Silva Filho²

¹Brazilian Aeronautics Institute of Technology—ITA, São José dos Campos, Brazil
²Santa Cecilia University—UNISANTA, Santos, Brazil
Email: mribeiro@ita.br, vdias@ita.br, inacio@unisanta.br

Received 26 February 2014; revised 20 March 2014; accepted 28 March 2014

Copyright © 2014 by authors and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/

Abstract

Nowadays, application model systems for decision-making based on non-classical logic such as Paraconsistent Logic are used successfully in the treatment of uncertainties. The method presented in this paper is based on the fundamental concepts of Paraconsistent Annotated Logic with annotation of 2 values (PAL2v). In this study, two algorithms based on PAL2v are presented gradually, to extract the effects of the contradiction in signals of information from a database of uncertain knowledge. The Paraconsistent Extractors Algorithms of Contradiction Effect-Para Extrcíś applied to filters of networks of analyses (PANets) of signal information, where uncertain and contradictory signals may be found. Software test case scenarios are subordinated to an application model of Paraconsistent decision-making, which provides an analysis using Paraconsistent Logic in the treatment of uncertainties for design software testing strategies. This quality-quantity criterion to evaluate the software product quality is based on the characteristics of software testability analysis. The Paraconsistent reasoning application model system presented in this case study, reveals itself to be more efficient than the traditional methods because it has the potential to offer an appropriate treatment to different originally contradicting source information.

Keywords
Paraconsistent Logic, Design Testing Strategies, Software Testability, Paraconsistent Decision Making Model

1. Introduction

This paper is the result of studies carried out for the Brazilian National Water Agency, within the scope of the Amazonian Integration and Cooperation Project for the Modernization of Hydrological Monitoring (in Portuguese, ICAMMH). The project was developed at the Brazilian Aeronautics Institute of Technology (ITA) [1].

http://dx.doi.org/10.4236/jsea.2014.75034
The purpose of this paper is to present an application of Artificial Intelligence techniques using Paraconsistent Logic in the treatment of uncertainties, for design software testing strategies, in terms that allow an analysis of a quality-quantity criterion to evaluate the software product quality based on the characteristics of software testability: ISO/IEC 25010:2011 [2] and heuristics of software testability by Bach [3].

During the software product development activity on the ICAMMH project, software product engineers used the quality management core assets, according to the software product development plan and a description of the particular software product to be built in Software Product Lines (SPL) [4] [5]. These planned activities carry out actions to program measurement of software quality and evaluate the application software testing standards. There is a large gap between the requirement that software quality measurement should be carried out and the standards and practices on how to carry out the measurements of software quality. In this context of the ICAMMH project, the uncertainty in decision making on the design software testing strategies, as regarding defect and faults origins of the software product, is rarely captured explicitly, in order to provide continuous quality improvement of the software product engineering, during the development process of the software.

The problem to be addressed is how to create a quality-quantity criterion that can measure continuous improvement actions during the process of software product development by SPL, in accordance to software quality standards. The specific approach focuses on software product quality by software testability: sub characteristics of the maintainability SQuaRE 25010:2011 [2] and heuristics of software testability of Bach [3].

Two research questions (RQ) were formulated:

RQ1. Is it possible to construct a Paraconsistent Analysis System Model in a way that facilitates the establishment of test criteria and quality performance of tests based on the quality-quantity criteria to evaluate the software product?

RQ2. Is it possible to develop an application with artificial intelligence tools to evaluate uncertainty in software testability criteria: 1) the measurements are feasible enough to allow testers diagnose techniques and methods; and 2) the practices and reachable test criteria of software product quality?

While RQ1 is addressed in the domain of Software Product Quality Engineering based on Paraconsistent Analysis Model for Uncertainty Treatment in Design Testing Strategies on SPL, RQ2 is focused on the interpretation results of Paraconsistent analysis from the quality-quantity characteristics of testability criteria corresponding to the software product quality standards application mapped.

This research provides important tools, in order to develop an application of Paraconsistent Annotated Logic (PAL) in designing software testing to act as powerful computational tools, dedicated to software testability analysis, and based on quality-quantity criteria without contamination by contradictory information signs.

This paper is organized in five sessions: Section 2 presents the fundamental theories; Section 3, the methodology for designing testing software model analysis, with Paraconsistent Logic (PL); Section 4, an application of the Paraconsistent Annotated Logic (PAL); Section 5, the conclusions reached.

2. Fundamentals Theories

2.1. Environment Software Testing for the ICAMMH Project

Eight software product lines were originally planned for the ICAMMH, but only one of them was actually available at the time of this study. Thus, only the Software Product Line 2 (SPL2) was utilized in experimental research. Requirements data and specifications design were collected from two different experiments and placed within the following software testing artifacts: test plan, test procedures, test cases, and test scenarios, as in IEEE Std. 829 [6]. This was done using functional and structural techniques, described in Perry [7], and defect prevention analysis, as in Sharma et al. [8].

The Software Test (ST) team used the following black-box techniques in SPL2 in the functional test: 1) Equivalence Class Partitioning; 2) Boundary Value Analysis, and 3) Cause Effect Graphing. This ST team prepared 256 test cases for the 8 usecases developed for SPL2. To obtain the software testing results, Test Cases (TC), as in [7], based on use case requirements, as in [9], they were manually performed for SPL2.

Different metrics were used to assess the SPL2, as seen in Table 1. In the first column, the indicators and metrics, in the second column, the calculation formulas. The third column displays the values calculated for the first Integration Test (IT) of SPL2. The Regression Testing (RT) results (1˚ RT, and 2˚ RT 3˚ RT) are shown in subsequent columns. Table 1 shows the primary metrics: 1) testing coverage percent use case by the PUC indi-
Table 1. Software Production Line 2—Derived Metrics and Indicators [Test Cases Have Found Defects—TCFD (*)].

<table>
<thead>
<tr>
<th>Indicators and Metrics</th>
<th>Formula</th>
<th>1° RT</th>
<th>1° RT</th>
<th>2° RT</th>
<th>3° RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing Coverage Percent Use Case (PUC)—indicator</td>
<td>(qty.) UC/total of UC</td>
<td>6/8 = 0.75 [75%]</td>
<td>8/8 = 1 [100%]</td>
<td>8/8 = 1 [100%]</td>
<td>8/8 = 1 [100%]</td>
</tr>
<tr>
<td>Defects Found by Software Test (DFST)—measure of efficacy</td>
<td>TCFD/total of TC</td>
<td>91/170 = 0.53 [53%]</td>
<td>66/235 = 0.28 [28%]</td>
<td>57/238 = 0.24 [24%]</td>
<td>60/256 = 0.23 [23%]</td>
</tr>
<tr>
<td>Defects Found by week (DEW)—indicator</td>
<td>TCFD/total of weeks</td>
<td>91/6 = 15.2 [75%]</td>
<td>66/6 = 16.5 [100%]</td>
<td>57/4 = 14.25 [100%]</td>
<td>60/3 = 20 [100%]</td>
</tr>
<tr>
<td>Effectiveness Defect Detection (EDD)—indicator</td>
<td>TCFD (n − 1) RT / TCFD (n) RT</td>
<td>N/A</td>
<td>91/66 = 1.38 [6%]</td>
<td>66/57 = 1.16 [42%]</td>
<td>57/60 = 0.95 [48%]</td>
</tr>
<tr>
<td>Effectiveness of Removal of Defects (ERD)—indicator</td>
<td>Remove TC defects No. RT/TCFD</td>
<td>N/A</td>
<td>4/66 = 0.06 [6%]</td>
<td>24/57 = 0.42 [42%]</td>
<td>29/60 = 0.48 [48%]</td>
</tr>
<tr>
<td>Percentage of Test Coverage (PTC)—metric efficiency</td>
<td>(qty.) TC exercised/total of TC</td>
<td>95/170 = 0.56 [56%]</td>
<td>194/235 = 0.83 [83%]</td>
<td>199/238 = 0.84 [84%]</td>
<td>215/256 = 0.84 [84%]</td>
</tr>
<tr>
<td>Density Efficiency Software Test (DEST)—indicator efficacy</td>
<td>TCFD/(qty.) TC exercised</td>
<td>91/95 = 0.95 [95%]</td>
<td>66/194 = 0.34 [34%]</td>
<td>57/199 = 0.29 [29%]</td>
<td>60/215 = 0.28 [28%]</td>
</tr>
</tbody>
</table>

Table 1 also shows the secondary metrics: 1) the percentage of effectiveness in detection of defects, calculated by the DFTS measure of efficacy, decreased with time; and 2) the percentage of efficiency in the execution of ST, calculated by PTC metric efficiency, which increased with time.

Also in Table 1, a comparison with RT (previous to the current one), it is observed that: 1) an efficient defect detection, calculated by the EDD indicator, decreased with time, and that the percentage of effectiveness of removal of defects, calculated by the ERD indicator, increased with time. The derived or secondary metrics present in Table 1, should provide the attributes of software product quality, as well as mechanisms to control the software process. The relationship between object-oriented design metrics and testability of classes uses on Table 1 as reference study by [9] [10].

2.2. The Use of Reference Standards for Testability Software

The theoretical approach used in this experimental research in terms of building components for a Paraconsistent quality-quantity analysis were based on the standards below.

The software product quality model SQuaRE 25010:2011 is composed of eight properties (Functional Suitability, Reliability, Performance Efficiency, Operability, Security, Compatibility, Maintainability, Transferability), which are further subdivided into sub-characteristics that can be measured internally or externally. Maintainability quality property refers to the product’s ability to be modifiable and changeable. Testability is one such sub-characteristic; the capability of the software is determined by a set of internal attributes that can be measured [2].

The ICAMMH Project also made use of the James Bach’s testability heuristics, to design internal facilitators for the testability of the software in the form of a set of quality attributes to describe the testability sub-characteristic, within the maintainability property of SQuaRE 25000:2011 standards. Khan and Mustafa [11] “extended James Bach internal facilitators to 2 (two) new approaches: external and environmental facilitators”.

2.3. Paraconsistent Logic and the Treatment of Uncertainties

Paraconsistent Logic (PL) belongs to a non-classical logic category and its main feature is the revocation of the principle of non-contradiction (see [12]-[15]). The pioneers of PL are J. Lukasiewicz and the Russian philosopher N. A. Vasiliev [11], who around 1910, simultaneously, but independently, suggested the possibility of the existence of a logic that did not use the principle of non-contradiction.

The initial systems of PL contained all logical levels such as propositional and predicate calculi as well as the Logic of Superior Order are due to N. C. A. da Costa (see [16]-[18]). PAL belongs to a family of PL and can be represented through a lattice of four vertices. These four vertices represent extreme logical states of propositions.
According to Da Costa et al. [19] [20], in PAL, the proposition \( P \) is accompanied with annotations; each annotation belongs to a finite lattice and attributes a logic value to the correspondent logical proposition. It is considered that each evidence degree attributes a logic value to the proposition that belongs to a group of values composed of the constants of annotation of the lattice \( \{ \top, \bot, F, t, \perp \} \) for which the following order relationship is defined: \( \perp < t, \bot < F, t < \top \) and \( F < \top \). PAL may be represented through a Hasse diagram in which the constants in the vertices of a lattice will show extreme logical states to the propositions sentence, where: \( T = \text{Inconsistent}, t = \text{True}, F = \text{False} \) and \( \bot = \text{Paracompleteness} \). The PAL language, the semantics of a complete set of connectives and axioms is found with details in Da Silva Filho et al. [21] and [22].

2.4. The Main Algorithms of the Paraconsistent Annotated Logic with Annotation of 2 Values—PAL2v

According to Da Silva Filho et al. [21] and [22] a representation of how the annotations, or evidences, express the knowledge about a certain proposition \( P \) can be obtained through PAL. This is done through a lattice on the real plane with pairs \( (\mu, \lambda) \) which are the annotations seen in Figure 1. In this representation an operator \( \neg \) is fixed, defined as follows: \( \neg |\tau| \rightarrow |\tau| \) where \( \tau = \{(\mu, \lambda) | \mu, \lambda \in [0, 1] \subset \mathbb{R}\} \).

If \( P \) is a basic formula then: \( \neg (\mu, \lambda) = (\lambda, \mu) \), where \( \mu, \lambda \in [0, 1] \subset \mathbb{R} \). The operator \( \sim \) stands for the “meaning” of the logical symbol of negation of the system to be considered. The \( P(\mu, \lambda) \), can be intuitively read: “It is assumed that \( P \)’s belief degree (or favorable evidence) is \( \mu \) and disbelief degree (or contrary evidence) is \( \lambda \). Thus, \((1, 0)\) intuitively indicates total belief, \((0, 1)\) indicates total disbelief, \((1, 1)\) indicates total inconsistency, and \((0, 0)\) indicates total paracompleteness (indetermination)” [21] [22].

This Paraconsistent system for the treatment of uncertainties using the Paraconsistent algorithm can calculate values that are representative of information signals. In a Paraconsistent analysis system for decision-making, the inputs \( \mu \) and \( \lambda \) are values contained in the closed interval between 0 and 1, which belong to the set of real numbers. These two values come from two or more information sources, which search for favorable or contrary evidences in respect to the same proposition \( P \). Since they originate from different sources, these values may be similar, thus representing consistency, or be different, representing a contradiction. The resulting Paraconsistent system is called basic Paraconsistent Analysis Node (PAN), according to Da Silva Filho et al. [21] [22].

The projects that utilize PANs may carry out information treatment through Paraconsistent analysis networking, without using signal adjustments, neither weight that may leave the processing dependent on external factors. With these results, a PAN may be connected to other PAN, forming a Paraconsistent Analysis Decision Network (PANNet) to estimate values and perform analysis of information in different conditions.

The special case where it is known initially where exist many contradictions attributed to certain evidence of Proposition \( A \) (\( Pa \)) should be considered. This weakens the evidence going on to another Proposition \( B \) (\( Pb \)). In this case, a special configuration is used, where output represented by an interval of evidence \( \phi_{\text{ext}} \) of the Paraconsistent Analysis Node (PAN) controls the analysis carried out by another PAN. This configuration, where three-dimensional analysis occurs, is called the Paraconsistent Cube Analyzer (PCA), according to Da Silva Filho et al. [20] [21].

3. Methodology for Designing Testing Software Model Analysis

The topology of PANNet is arranged in order to treat uncertainties; it can be built with several interlinked PAN-algorithms. Thus, in a basic configuration, the preliminary analysis in a PAN produces a value of real Degree Certainty \( D_{CR} \) and a signaled interval of certainty \( \phi_{e}(\pm) \), referring to a unique proposition. According to Da Silva Filho et al. [23] [24] the Paraconsistent Algorithm Extractor of Contradiction effects (\( \text{ParaExtr}_{ea} \)) is applied to connections of the PANNet. This configuration forms a Paraconsistent Analysis Network, which is capable of extracting the effects of the contradiction, gradually from the signals of information that come from the uncertain knowledge database source. Figure 2 shows the representation of the algorithm extractor of contradiction effects that uses a network of three PANs.

The Paraconsistent Analysis for Extraction of Contradiction effects (\( \text{ParaExtr}_{ea} \)) algorithm is based on a topology that enables PANNet to represent a range of situations, where the existing (\( \phi_{e}(\pm) \)) group of clusters of similar semantic nature elements coming from several sources of information representing a modeling PAN Kernels, according to Da Silva Filho et al. [23] [24].

\( \text{ParaExtr}_{ea} \) — Extractor of Contradiction Effects Algorithm—Part (a), similar semantic nature
Figure 1. Organization associated lattice of the paraconsistent annotated logic [22].

Figure 2. The Paraconsistent algorithm extractor of contradiction effects (ParaExtrct)
1) Present $n$ values degrees of evidence that compose the group under study

$$G_\mu = (\mu_A, \mu_B, \mu_C, \cdots, \mu_n)$$

$0 \leq \mu \leq 1$.

2) Select the largest value among the degrees of evidence the group under study

$$\mu_{\text{max}, A} = \max \left( \mu_A, \mu_B, \mu_C, \cdots, \mu_n \right).$$

3) Consider the largest value among the degrees of evidence the group under study in the favorable degree of evidence $\mu_{\text{max}, A} = \mu_{\text{sel}}$.

4) Consider the smallest value among the degrees of evidence the group under study in the favorable degree of evidence:

$$\mu_{\text{min}, A} = \min \left( \mu_A, \mu_B, \mu_C, \cdots, \mu_n \right).$$

5) Transform the smallest value among the degrees of evidence the group under study in the unfavorable degree of evidence

$$\lambda_{\text{sel}} = 1 - \mu_{\text{min}, A}.$$ 

6) Do the Paraconsistent analysis among the selected values.

$$\mu_{R1} = \mu_{\text{sel}} \circ \lambda_{\text{sel}} \text{ where } \circ \text{ is a Paraconsistent action of the PAN kernel}.$$

7) Increase the obtained value $\mu_{R1}$ in the group under study, excluding the two values $\mu_{\text{max}}$ e $\mu_{\text{min}}$ from this, selected previously.

$$G_\mu = (\mu_{G1}, \mu_A, \mu_B, \mu_C, \cdots, \mu_n) - (\mu_{\text{max}}, \mu_{\text{min}}); \text{ where } i = A, \cdots, N$$

8) Return to item 2 until the Group under study has only 1 element resulting from the analyses,

Go to item 2 until $G_\mu = (\mu_{\text{cf}})$.

**Function PAN Kernel ($\mu, \lambda$)**

1) Enter with the input values

$\mu$ /*favorable evidence Degree $0 \leq \mu \leq 1$

$\lambda$ /*unfavorable evidence Degree $0 \leq \lambda \leq 1$

2) Calculate the normalized Contradiction Degree

$$\mu_{\text{ctr}} = \frac{\mu + \lambda}{2}.$$

3) Calculate the interval of evidence resulting

$$\varphi_{\text{eint}} = 1 - |2\mu_{\text{ctr}} - 1|.$$

4) Calculate the Certainty Degree

$$\text{Dc} = \mu - \lambda.$$

5) Calculate the Contradiction Degree

$$\text{Dct} = \mu_{\text{ctr}} + 1.$$

6) Calculate the distance d into Lattice

$$d = \sqrt{((1 - |\text{Dc}|)^2 + \text{Dct}^2)}.$$

7) Compute the output signal

If $\varphi_{\text{eint}} < 0$, 25 OR $d > 1$, then

do: $S1 = 0.5$ e $S2 = \varphi e(\pm)$:

uncertainty go to item 11

Else: go to next step

8) Calculate the real Certainty Degree

If $\text{Dc} > 0$ then $G_{\text{cr}} = (1 - d)$

If $\text{Dc} < 0$ então $G_{\text{cr}} = (d - 1)$

9) Calculate the real Evidence Degree

$$\mu_{\text{er}} = \frac{G_{\text{cr}} + 1}{2}.$$

10) Present the outputs

$S1 = \mu_{\text{er}}$ and $S2 = \varphi e(\pm)$

11) End}

The new algorithm $\text{(ParaExtrctrPlus)}$ is based on a topology that enables PANNet to represent a range of situations, where the existing $(\varphi_{\text{ctr}})$ group with possibility to represent also clusters, with different semantic nature elements coming from several sources of information representing a modeling PCA Kernels, As exposed below.

$\text{ParaExtrctrPlus} — \text{Extractor of Contradiction Effects plus Algorithm—Part (b), different semantic nature}$
1) Present $n$ values degrees of evidence that compose the group under study

$$G_\mu = \left( \mu_A, \mu_B, \mu_C, \cdots, \mu_n \right) \ast / \text{degrees of evidence} \ 0 \leq \mu \leq 1 \ast / .$$

2) Select the largest value among the degrees of evidence the group under study

$$\mu_{\text{max}, A} = \max \left( \mu_A, \mu_B, \mu_C, \cdots, \mu_n \right).$$

3) Consider the smallest value among the degrees of evidence the group under study in the favorable degree of evidence:

$$\mu_{\text{min}, A} = \min \left( \mu_A, \mu_B, \mu_C, \cdots, \mu_n \right).$$

4) Consider the smallest value among the degrees of evidence the group under study in the favorable degree of evidence:

5) Transform the smallest value among the degrees of evidence the group under study in the unfavorable degree of evidence.

$$\text{If } \mu_{\text{min}, A} = \mu_{\text{source}} \text{ then do } \lambda_{\text{sel}} = \lambda_{\text{min}, A};$$

/* if there is more than one $\mu_{\text{min}, A} = \mu_{\text{source}}$, choose the largest $\lambda$ corresponding to $\mu_{\text{min}, A}$ */

6) Do the Paraconsistent analysis among the selected values

$$\mu_{\text{R1}} = \mu_{\text{sel}} \Diamond \lambda_{\text{sel}}; \ */ \text{ where } \Diamond \text{ is a Paraconsistent action of the PCA Kernel } */$$

7) Increase the obtained value $\mu_{\text{R1}}$ in the group under study, excluding from this the two values $\mu_{\text{max}} e \mu_{\text{min}}$, selected previously

$$G_{\mu} = \left( \mu_{\text{R1}}, \mu_A, \mu_B, \mu_C, \cdots, \mu_n \right) - \left( \mu_{\text{max}}, \mu_{\text{min}} \right); \text{ where } i = A, \cdots, N .$$

8) Return to item 2 until the Group under study has only 1 element resulting from the analyses.

Go to item 2 until $G_{\mu} = \left( \mu_{\text{R1}} \right)$.

**Function PCA Kernel** ($\varphi_{\text{ext}}, \mu, \lambda$) {

1) Enter with the input values

$\mu /*$ favorable evidence Degree $0 \leq \mu \leq 1$

$\lambda /*$ unfavorable evidence Degree $0 \leq \lambda \leq 1$

$\varphi_{\text{ext}} /*$ external interval Evidence*/

2) Verify initial conditions

If $I \varphi_{\text{ext}} < 0, 25$, then
do: $S1 = 0.5 e S2 = \varphi_{\text{ext}}$: uncertainty go to item 15
Else: go to next step

3) Calculate the max value of Evidence degreetending to max (true, false)

$$\mu_{\text{Emax,t}} = \left( 1 + \varphi_{\text{ext}} \right)/2$$

$$\mu_{\text{Emax,f}} = \left( 1 - \varphi_{\text{ext}} \right)/2$$

4) Calculate the Evidence degree internal results $\mu_{\text{eint}} = (\mu - \lambda + 1)/2$

5) Verify the conditions

If $\mu_{\text{eint}} \geq \mu_{\text{Emax,t}}$, then do: $D_c = \varphi_{\text{ext}}$, $Dct = 1 - \varphi_{\text{ext}}$, and $\varphi_{\text{ext}} = \varphi_{\text{e}} (+)$ go to item 12;

If $\mu_{\text{eint}} \geq \mu_{\text{Emax,f}}$, then do: $D_c = -\varphi_{\text{ext}}$, $Dct = 1 - \varphi_{\text{ext}}$, and $\varphi_{\text{ext}} = \varphi_{\text{e}} (+)$ go to item 12; else next item.

6) Calculate the normalized Contradiction Degree $\mu_{\text{cr}} = (\mu + \lambda)/2$

7) Calculate the interval of Evidence resulting

$$\varphi_{\text{eint}} = 1 - \left( 2\mu_{\text{cr}} - 1 \right)$$

8) Calculate the Certainty Degree

$D_c = \mu - \lambda$

9) Calculate the Contradiction Degree

$Dct = \mu - \lambda + 1$

10) Calculate the distance d into Lattice

$d = \text{root}( (1 - |D_c|)^2 + Dct^2) )$

11) Compute the output signal

if $\varphi_{\text{eint}} < 0, 25$ OR $d > 1$, then
do: $S1 = 0.5 e S2 = \varphi_{\text{eint}} (+)$: uncertainty go to item 15
Else: go to next step

12) Calculate the real Certainty Degree

If $Dc > 0$ then $Dcr = (1 - d)$
If \( D_c < 0 \) then \( D_{cr} = (d - 1) \)

13) Calculate the real Evidence Degree

\[ \mu_{cr} = \frac{(D_{cr} + 1)}{2} \]

14) Present the outputs

\( S_1 = \mu_{cr} \) and \( S_2 = \phi_e(\pm) \)

15) End

Structural metrics are based on the properties of flow graph models of programs. The attention is focused on control-flow and data-flow complexity. Linguistic complexity is ignored. Cyclomatic Complexity \( V(G) \) is a measure of the complexity of a module’s decision structure. It is the number of linearly independent paths and therefore, the minimum number of paths that should be tested. Invented by Thomas McCabe (1974) to measure the complexity of a program’s conditional logic: counts the number of decisions in the program, under the assumption that decisions are difficult for people and makes assumptions about decision-counting rules and linear dependence of the total count to complexity. McCabe’s Cyclomatic complexity is defined as \( V(G) = e - n + 2p \), where:

- \( e \) = number of links in the flow graph;
- \( n \) = number of nodes in the flow graph; and
- \( P \) = number of disconnected parts of the flow graph.

The ParaExtrcr and ParaExtrcrPlus algorithms are made using C programming language and both modules PAN and PCA Kernel. Control flow graphs describe the logic structure of software modules. A module corresponds to a single function or subroutine in typical languages, has a single entry and exit point, and is able to be used as a design component via a call/return mechanism. Each flow graph consists of nodes and edges. The nodes represent computational statements or expressions, and the edges represent transfer of control between nodes. As seen Figure 3.

The complexity of several graphs considered together is equal to the sum of the individual complexities of those graphs. The strategy of modularization ParaExtrcr and ParaExtrcrPlus algorithms aims to implement C language source code programs for optimal number of cyclomatic complexity for each PAN and PCA Kernel function around values lower than 10. As seen Figure 4 and Figure 5.

Practical Software Quality Issues where considered: 1) no program module should exceed a cyclomatic complexity of 10; and 2) software refactoring are aimed at reducing the complexity of a program’s conditional logic. Essentially, with cyclomatic complexity, higher numbers are “bad” and lower numbers are “good”. We use cyclomatic complexity to get a sense of how hard any given code may be to test, maintain, or troubleshoot as well as an indication of how likely the code will be to produce errors.

**Converting Code to Graph**

"Minimum number of tests necessary to cover all edges is equal to the number of independent paths through the control-flow graph"

- A graph is strongly connected if for any two nodes \( x, y \) there is a path from \( x \) to \( y \) and vice versa
- A circuit is a path that begins and ends at the same node
- A cycle is a circuit with no node (other than the starting node) included more than once

**Figure 3. Strategies to Convert Code to Graph (McCabe V(g)).**
4. Application of the PAL2v for Design Testing Software and Discussion

Developers at ITA Project were also responsible for ensuring that the testability issues were addressed in the
requirements and design phases of development to support test planning and test design. Testers interacted with a team of designers to deal with the occurrence of testability problems, as well as the documentation of the design specifications used as the basis of test cases that would support integration testing and unit testing. The master test plan was revisited and enhanced with new information as needed upon every release of every SPL.

The ITA Software Testing (ST) team consisted of four (4) members with the following responsibilities for profile: senior test leader, usability test engineer, manual test engineer and automated test engineer. According to Elfriede [25] the capabilities of the ST team may affect the success or failure of the testing effort. It is not enough for a testing team to be technically proficient with the testing techniques and tools necessary to carry out the actual tests. Depending on the complexity of the domain, a test team should also include members who have a comprehensive understanding of the problem domain.

Using the above standards in SPL2, a treatment system of uncertainty for decision-making based on the PAL2v, was applied to investigate how to treat degrees of evidence related to the use for designing testing techniques based on software testability by quality-quantity indicators. Bach’s quality facilitator heuristics for software testability were matched to the testability quantity indicators of SQuaRE 25010 to analyze the impact on best practices and techniques the test scenarios analysis.

Considering two information sources, [A] and [R], that send the evidence concerning a certain Proposition \( P_n \) to the analysis and decision-making system, as seen in Figure 6 for the 1st step—(primary sources of information: accept or reject the acquisition data). The data: [A] — acceptance — degree of acceptance value of the proposition in the interval \([0,10]\); [R] — rejection — degree of rejection value of the proposition in the interval \([0,10]\).

The degrees of evidence acquisition of the samples were obtained through mathematical equations, through normalization from modelling evidence signals, by acceptance and rejection acquisition data, with the exposed values in the degrees of evidence valued between 0 and 1 in the PAL2v lattice.

The degree of favorable evidence (\( \mu \)), expressing the level of acceptance of a statement \( P_i \), and the degree of unfavorable evidence (\( \lambda \)), expressing the level of non-acceptance, were defined by responses to each object proposition [statements \( (P_1,\ldots,P_7) \)]. The values were calculated respectively according to the following Relative Strength Index (RSI) formula in Figure 7, for the 2nd step it should begin with the sentence “RSI function PAL2v modeling Paraconsistent proposition \( P_n \)”, resulting in a degree of individual evidence from each tester (3rd step).

As seen Figure 8, in the 4th step, the Paraconsistent Extractor of Contradiction Effects (\( ParaExtr_{\text{cr}} \)) Algorithm is composed by connections among PANs. This configuration forms a Paraconsistent network capable to extract the effects of the contradiction, in gradual way of the signals of information, from resulting degree of individual evidence of heuristic of software testability facilitator (\( \mu_{\text{erf}} \)).

The degree of evidence \( \mu_{\text{erf}} \) is a single value corresponding to a tester group (I = 1, 2, 3, 4) opinion for each Bach testability heuristics facilitator. Through successive Paraconsistent analyses, the effects of the contradictions are extracted and presented. As seen Figure 8, in the 4th step and Table-A.

Thus, contradictions (\( \mu_{\text{cr}} \)) in certain proposition analyses may affect the value of degree of certainty supplied by the analysis of an Object Proposition (\( P_1,\ldots,P_7 \)). This PANNet has a process of normalization where each PAN output, through to \( ParaExtr_{\text{cr}} \) Algorithm, represents a Degree of Resultant Real Evidence value \( \mu_{\text{erf}} \) accompanied by a Resultant Interval of Certainty value \( \phi_{\text{eint}} \), as seen Figure 8, in Table-A.

Three basic rules may be considered to make the combinations of the results in Paraconsistent Analysis network:

1) Proposition analyzed in the PANs maybe logically combined through the Degrees of resultant Real Certainty originated from the analysis, and thus make Paraconsistent and thus make different interconnections in the Paraconsistent Analysis Network;

2) The values of the Degrees of Resultant Real Certainty, as well as the intervals of Real certainty originated from the PANs, referring to the different propositions, may be logical treated by conjunction (AND) and disjunction (OR) or algebraically by addition or subtraction of their values, according to the characteristics and topology of the PANNet project;

3) The values of the Degrees of Resultant real certainty may be transformed, by normalization, into values between 0 and 1, in the real number interval, and thus considered as Degrees of Evidence of other propositions which are being analyzed by other different PANs. In this way, the interconnections among the PANs, will be done through the analysis of the evidences.
Figure 6. Bach’s testability heuristics associated to a SQaRE 25010 with the acquisition data for the results of the degree of acceptance or rejection for each proposition.

The relations between the resulting interval of certainty value $\varphi_e$ and normalized degree of contradiction $\mu_{\text{ctr}}$ value show how the level of contradiction in the analysis of PANNet. When taken together, they will promote evidence to make a decision that results in higher accuracy and reliability, as seen in Table 2.

It was verified from the characteristic line on the graph $[\varphi_e \text{ versus } \mu_{\text{ctr}}]$ from data in Table 2: that when the resultant interval of evidence is maximum value $\varphi_e = 1$ and the normalized degree of contradiction value $\mu_{\text{ctr}} = 0.5$, represents a null contradiction.

Maximum contradictions, represented by a value of null resultant Interval of evidence, occur in two values of degree of contradiction $\mu_{\text{ctr}}$: 1) when the normalized degree of contradiction is equal to zero, indicating maximum contradiction by indetermination ($\perp$), and 2) when the normalized degree of contradiction is maximum, and indicating maximum contradiction by inconsistency ($\top$).

In this situation there will be two conditions: 1) if the unfavorable evidence $\lambda$ is greater than the favorable evidence $\mu$, the resultant interval of evidence $\varphi_e$ may present values lower than 0.5 and greater or equal to zero,
Figure 7. The RSI oscillator function modelling degree of evidence the PAL2v lattice.

Table 2. Contradiction Analysis—Lattice PAL2v.

<table>
<thead>
<tr>
<th>Resultant Interval of Evidence ($\phi_e$)</th>
<th>Normalized degree of Contradiction ($\mu_{ctr}$)</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.000</td>
<td>Indetermination ⊥</td>
</tr>
<tr>
<td>0.25</td>
<td>0.125</td>
<td>boundary limit ⊥</td>
</tr>
<tr>
<td>0.50</td>
<td>0.250</td>
<td>Data valid</td>
</tr>
<tr>
<td>0.75</td>
<td>0.375</td>
<td>Data valid</td>
</tr>
<tr>
<td>1.00</td>
<td>0.500</td>
<td>Null contradiction</td>
</tr>
<tr>
<td>0.75</td>
<td>0.625</td>
<td>Data valid</td>
</tr>
<tr>
<td>0.50</td>
<td>0.750</td>
<td>Data valid</td>
</tr>
<tr>
<td>0.25</td>
<td>0.875</td>
<td>boundary limit ⊤</td>
</tr>
<tr>
<td>0.00</td>
<td>1.000</td>
<td>Inconsistency ⊤</td>
</tr>
</tbody>
</table>

and 2) if the favorable evidence $\mu$ is greater than the unfavorable evidence $\lambda$ the resultant interval of evidence $\phi_e$ may present values greater than 0.5 and lower or equal to zero.

The Table 3 shows the characteristics found of the resultant interval of evidence $\phi_e$ related with the values of

\[ \mu = \left(1 - \left(1 - \frac{1}{\lambda+1}\right)^{AF}\right) \left(1 - \left(1 - \frac{1}{\mu+1}\right)^{AF}\right) \]

This formula was designed by Welles Wilder [1978], the RSI (Relative Strength Index) is an oscillator that allows to compare the average magnitude of gains from the average of the losses on a scale interval.
resultant degree of real evidence $\mu_{erf}$.

In the characteristic segment line on the graph $[\phi_e$ versus $\mu_{erf}]$ from data in Table 3 enables to verify that the resultant interval of evidence $\phi_e$ represents the permitted value for variation of resultant degree of real evidence $\mu_{erf}$, in situation of the normalized degree of contradiction value $\mu_{ctr}$.

The resultant interval of evidence $\phi_e$ impacts on the maximum of the output degree of evidence tending to logical state True or False, as seen in Figure 9.

In the end of the analysis, the ParaExtr$_{cr}$ algorithm presents as result a real evidence degree values $\mu_{erf}$ representative of the evidence degree group, was validated in agreement, the interval of contradiction analysis, output values from 4th step, according to Table 2: between boundary limit indetermination ($\perp$) and boundary limit inconsistency or paracompleteness (T).

When viewing Table-A of results for each proposition $P_n$ of heuristics of software testability in the 4th step, in Figure 8, the small value $\mu_{erf} = 0.38$, refers to the operability facilitator. This indicates a potential deficiency in the training of the technical staff of software testing, compromising the quality of the software product through poor implementation of best practices for testing and lack of operational experience.

Another of Bach’s facilitators that showed medium results was controllability, with the value $\mu_{erf} = 0.508$. This means there were problems in the process of operating a system or software components under specified conditions of testing coverage. Some strategies that apply in test case scenarios with black box techniques: Equivalence Class Partitioning, Boundary Value Analysis, and State Transition Table, have shown to be inconsistent and partly failing.
Table 3. Characteristics resultant degree evidence—PAL2v Lattice.

<table>
<thead>
<tr>
<th>Resultant Interval of Evidence ($\varphi_e$)</th>
<th>Resultant degree evidence ($\mu_{erf}$)</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.000</td>
<td>False $F$</td>
</tr>
<tr>
<td>0.25</td>
<td>0.125</td>
<td>boundary limit $F$</td>
</tr>
<tr>
<td>0.50</td>
<td>0.250</td>
<td>Data valid</td>
</tr>
<tr>
<td>0.75</td>
<td>0.375</td>
<td>Data valid</td>
</tr>
<tr>
<td>1.00</td>
<td>0.500</td>
<td>Indetermination $\bot$</td>
</tr>
<tr>
<td>0.75</td>
<td>0.625</td>
<td>Data valid</td>
</tr>
<tr>
<td>0.50</td>
<td>0.750</td>
<td>Data valid</td>
</tr>
<tr>
<td>0.25</td>
<td>0.875</td>
<td>boundary limit $t$</td>
</tr>
<tr>
<td>0.00</td>
<td>1.000</td>
<td>True $t$</td>
</tr>
</tbody>
</table>

The stability facilitator, in Figure 8, showed better numbers, it had a high value of $\mu_{erf} = 0.602$ and null contradiction. This facilitator showed a detection of defects with a satisfactory level in each release of the Software Product Line (SPL). It represents the changes in Test Cases (TC) and testing techniques employed by the software test team to find bugs which were assimilated incrementally with efficiency.

The 5th step is “go to the Paraconsistent analysis with the global evidence degree (filter) pattern”, uses the ParaExtrcrPlus algorithm in order to homogenize different facilitators, which is key to achieving an overall proposition concerning an object: $P_0$—“It is possible to matched quantity testability indicators of SQuaRE 25010 with the quality facilitators heuristics of software testability by James Bach, to analyze the impact of design testability using best practices and techniques on test scenarios”.

The ParaExtrcrPlus algorithm is a network for treatment of uncertainties, which can be built with several interlinked PCA-algorithms. The hypothesis of extraction of the effects of the contradiction from several non-homogeneous sources has the principle that: 1) if the first treated signals are the most contradictory, the impact each propagation of resultant Interval of Evidence must be analyzed, subsequently to another PCA kernel; and 2) gradually filters inconsistencies to treat the effects of contradiction in information from each resultant real degree evidence through the PANNet to obtain the influence on global final results.

In the 5th step, the ParaExtrcrPlus algorithms to produce the global resultant real degree of evidence $\mu_{erf}$ is equal to 0.537, where $\mu_{erf} = 0.511$ and $\varphi_e$ is positive near to value 1, representing a tendency of null contradiction. As seen in Table B, Figure 8.

The extraction process reduces the effects of the inconsistencies and it presents as answer a closer representative value of the reality, through the signals of contradiction by evidence of their ranges to reduce the partial contradiction $\mu_{erf}$ between the different Bach's testability facilitators.

In each analysis carried out the PAN interlinks of PANNet, PCA or PAN Kernel, the Interval of certainty is
available, besides the Resultant Degree of Evidence according to Table 3. The two values will allow an identification of the condition for analysis of uncertainty of each proposition in the PAN Nettopology. With the value of Interval of Certainty, there exists an indication of where the system may react. Whether reduce or strengthen the evidences with the objective of diminishing the conflicts and increasing of Degree of Certainty.

The level of demand for the system makes the decision favorable; in other words, the final result \( \mu_{ef} \) is “true” and satisfactory for the global proposition object \( P_0 \). The quality-quantity mapping [modelling] standards of Bach’s testability model [3] and SQuaRE 25010 [2], provided an integrated view of complementary mechanisms to improve best practices, methods and techniques results on the diagnostic analysis for designing testability of software adopted by ICAMMH project Test Team.

5. Conclusions
This paper described the design characteristics of the PANNet topology carrying out different connections, which provided result as a real evidence degree representative on PAL2v lattice. The results demonstrate the feasibility of the quality-quantity process model for measure software testability.

The \( \text{ParaExtr}_{cr} \) algorithms presented are capable of extracting contradiction effects from groups of evidence signals regarding quality-quantity propositions of testability through basic concepts of Paraconsistent Annotated Logic for two values (PAL2v).

The modelling through PANNet topology combining PAN and PCA kernels of \( \text{ParaExtr}_{cr} \) algorithms have provided safe information about propositions with a higher or lower degree of contradiction. With this information the system is able to make more reliable decisions, besides having the values to act on the input signal control, weakening or strengthening evidences, in order to reduce contradiction effects.

The \( \text{ParaExtr}_{cr} \) and the \( \text{ParaExtr}_{crPlus} \) algorithms have proposed new structures and different configurations of Paraconsistent analyses networks that were designed for treatment of uncertainties for designing software testing strategies. In that context the new approach aims to improve quality assurance for the development of software products.

Testability is a key factor in the quality of the software product, to propose new approaches to verifying and validating the quality for design strategies for software, during the development cycle.

This paper demonstrated an effective improvement in testability analysis and software because with the use of PAL2v algorithms, it presents a new way to treat the effects of gradually contradiction in information, also provides monitoring conditions, and adjustments in steering of the process.

The results obtained in this paper bring contributions to build robust Paraconsistent Specialists Systems, to act as powerful computational tools dedicated to software testability analysis without contamination by contradictory information signals.

Acknowledgements
The Authors would like to thank the Brazilian Aeronautics Institute of Technology (ITA), for its scientific development incentives by using Research & Development Projects at ITA; the Brazilian National Water Agency (ANA), for the opportunity of participating in its Project; the Finance Agency for Studies and Projects (FINEP); and also the Casimiro Montenegro Filho Foundation (FCMF) for its available infrastructure.

References


