

# A Set of Empirically Validated Development Guidelines for Improving Node-RED Flows Comprehension

Diego Clerissi<sup>2</sup>, Maurizio Leotta<sup>1</sup> <sup>a</sup> and Filippo Ricca<sup>1</sup>

<sup>1</sup>Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), Università di Genova, Italy

<sup>2</sup>Dipartimento di Informatica, Sistemistica e Comunicazione (DISCO), Università di Milano, Bicocca, Italy

**Keywords:** Node-RED, Guidelines, Comprehension, IoT Web Based Systems, Visual Development.

**Abstract:** Internet of Things (IoT) systems are rapidly gaining importance in the human society, providing a variety of services to improve the quality of our lives, involving complex and safety-critical tasks; therefore, assuring their quality is of paramount importance.

Node-RED is a Web-based visual tool inspired by the flow-based programming paradigm, built on Node.js, and recently emerged to support the users in developing IoT systems in a simple manner. The community behind Node-RED is quite active and encourages artefacts sharing. Thus, the Node-RED flows developed and submitted to public usages should be easy to comprehend and integrate within already existing systems, also in preparation of future maintenance and testing activities. Unfortunately, no consolidated approaches or guidelines to develop comprehensible Node-RED flows currently exist.

In this paper, we propose a set of guidelines to help the Node-RED developers in producing flows that are easy to comprehend and use. We have designed and conducted an experiment to evaluate the effect of the guidelines in Node-RED flows comprehension. Results show that the adoption of the guidelines significantly reduces the number of errors (p-value = 0.00903) and the time required to comprehend Node-RED flows (p-value = 0.04883).

## 1 INTRODUCTION

In the context of the Internet of Things (IoT), Node-RED has become a practical solution to easily develop and share IoT systems.


Node-RED is a Web-based tool inspired by the flow-based programming paradigm (Morrison, 2010) and built on top of the Node.js framework. In Node-RED, a *node* represents part of a service logics and is largely configurable, while a *flow* describes the way the nodes collaborate and communicate.

Daily, nodes and flows are developed and uploaded to the Node-RED library by the developers participating in the community (over 2000 nodes in 2019<sup>1</sup>), as solutions to general or specific problems, and anyone may download part of this content to integrate it within existing systems. Nodes can execute a variety of tasks, like reading values from a database, running a JavaScript function, receiving the feeds from a Twitter

account, establishing a communication between two devices using the MQTT protocol, and more.

As the flow-based programming paradigm prescribes (Morrison, 2010), nodes are black-box components that hide all the implementation details (i.e., basically, JavaScript functions and graphical features). The developer can select the nodes she desires and wire them together in order to implement the system she wants, without having the complete knowledge of the nodes implementation she is using.

As any other programming tool and language, Node-RED lets the developer to choose her own *programming style* while implementing new nodes and flows. Since Node-RED is a visual tool, along with the programming style, there is also the **comprehensibility factor** related to the *graphical style* adopted for wiring the nodes together to compose the Node-RED flows, as well as for carefully choosing meaningful names for the nodes; in general, this is a problem more frequently found at design stage. The lack of a disciplined approach as a guidance for developing Node-RED flows could result in messy “spaghetti” artefacts, very hard to comprehend and use, which may produce

<sup>a</sup>  <https://orcid.org/0000-0001-5267-0602>

<sup>1</sup><https://flows.nodered.org/>

unexpected outcomes when they are integrated into further complicated systems, without mentioning the pain of maintaining and testing them.

Up to now, no consolidated approaches supporting Node-RED developers in producing reusable and comprehensible flows exist, and only few basic and unofficial attempts proposing best practices and design patterns have been made so far<sup>2</sup>.

In this paper, we propose a preliminary set of guidelines to produce Node-RED flows that are easy to comprehend by construction, and more suitable to reuse, maintenance and test. We have investigated the benefits of adopting our guidelines by means of an experiment involving ten master students, where two selected Node-RED systems, each one developed with and without our guidelines, are compared in a comprehension scenario.

In Section 2 the guidelines are described, while some Node-RED comprehensibility issues pertaining the two selected Node-RED systems are introduced in Section 3, showing how the guidelines can be applied on them. The experiment and the results are discussed in Sections 4 and 5, respectively. Finally, related work are presented in Section 6, and the conclusions are discussed in Section 7.

## 2 PROPOSED GUIDELINES

The guidelines we propose in this paper address some common Node-RED comprehensibility issues, which may emerge while developing flows or trying to understand and integrate flows provided by an external source (e.g., the Node-RED community library). Issues may concern confusing nodes names, hidden loops and loss of messages, lack of conditional statements, unexpected inactive nodes, and more. More details about issues are provided in Section 3.3. The guidelines aim at supporting Node-RED developers in producing flows that are easy to comprehend by construction, and suitable for future reuse, maintenance and testing activities.

The guidelines have been inspired by several design works addressing systems quality using UML and BPMN (Ambler, 2005; Mendling et al., 2010; Unhelkar, 2005; Reggio et al., 2011; Reggio et al., 2012), and by our experience in IoT systems design and Node-RED flows development (Clerissi et al., 2018; Leotta et al., 2018). UML is one of the most used notational languages (Reggio et al., 2014), and differs to Node-RED in many aspects: while UML works at

design level and describes the static and dynamic details of a system, Node-RED is an executable visual language used to implement, execute and deploy a working system. The constructs they use are quite different, as well as their syntax and semantics. Nevertheless, we have experimented in practice that some design and technology-independent principles can be inherited from UML even to solve specific Node-RED issues (Clerissi et al., 2018; Leotta et al., 2018).

To better comprehend the Node-RED terminology and the issues that our guidelines try to address, in Table 1 we recap a short list of terms and definitions, extracted and elaborated from the Node-RED official documentation<sup>3</sup>.

The guidelines we propose can be classified into four types, based on the comprehensibility issues they address: **Naming**, **Missing Data**, **Content**, and **Layout**.

### 2.1 Naming

**Node Name Behaviour (NNB).** Each Node-RED node should have a unique (unless a duplicate of another existing node) and meaningful name, suggesting its high-level behaviour (Lange et al., 2006). The name of a node should make explicit the *action(s)* performed by the node and the *object(s)* receiving such *action(s)*. An *object* may refer to a message property or a global/flow variable, written in upper-case to be more visible within the flow (Reggio et al., 2012).

**Flow Name Behaviour (FNB).** Each Node-RED flow should have a unique and meaningful name, summarizing in a very concise way its high level behaviour (Lange et al., 2006).

### 2.2 Missing Data

**Node Effective Contribution (NEC).** By adapting to Node-RED the terms used by Ambler (Ambler, 2005), there should neither exist *black hole* nodes nor *miracle* nodes. A black hole node is a node with no leaving wires but output pins  $> 0$ , which means that the node output might be lost or unused by the flow, while a miracle node is a node with no entering wires but input pins  $> 0$ , which means that the node cannot be explicitly activated or is missing some data.

**Conditions Consistency and Completeness (CCC).** The conditions of every switch node (i.e., a core Node-RED node basically implementing the switch/if constructs of every programming language, and used to route the messages by evaluating a set of conditional

<sup>2</sup><https://medium.com/node-red/node-red-design-patterns-893331422f42>

<sup>3</sup><https://nodered.org/docs/>

Table 1: Node-RED Essential Terms and Definitions.

Term	Definition
Node	The basic Node RED component, representing (part of) the logics of a service/functionality. Each node has a type describing its general behaviour and a set of custom properties.
Flow	The logical way the nodes are wired, expressing how they collaborate by exchanging messages.
Sub-Flow	Each self-contained logical portion of a flow, contributing to its completion.
Wire	The edge used to graphically connect two nodes in a flow.
Pin	The input/output port of a node where a wire enters/leaves.
Message	A data object exchanged by some nodes, characterized by a sequence of configurable properties.
Global/Flow Variable	A variable defined in a node and visible by all the flows or by just the one containing that node.

statements over global/flow variables or message properties<sup>4</sup>) should not overlap and be complete (i.e., their disjunction returns true) (Ambler, 2005; Reggio et al., 2012), in order to handle separately all the possible scenarios.

### 2.3 Content

**Sub-Flows Relatedness (SFR).** The sub-flows composing a flow should be logically related among each others (Lange et al., 2006), following the design principle of high cohesion and low coupling (Martin, 2003). Two sub-flows  $F_1$  and  $F_2$  are logically related if, e.g.:

- $F_1$  and  $F_2$  describe (part of) the behaviour of the same device;
- $F_1$  and  $F_2$  contribute to the same service/functionality;
- $F_1$  and  $F_2$  share some variables or other data;
- $F_1$  is activated by  $F_2$  or vice versa;

**Flow Content (FC).** If a flow is overpopulated, its content should be simplified (Mendling et al., 2010; Unhelkar, 2005; Ambler, 2005), by identifying some of its sub-flows, and either: (a) physically split and connect them together through link nodes (i.e., a core Node-RED node used to add a virtual wire between two sub-flows<sup>5</sup>), or, (b) collapse them into corresponding sub-flow nodes (i.e., a core Node-RED node used to collect sub-flows to favour reuse and reduce layout complexity<sup>6</sup>). Since Node-RED flows design and development phases are strongly related due to the visual nature of the tool, as in the case of more general design activities, there is a positive correlation between flows size and complexity. A flow is then classified as overpopulated if the number of nodes it contains is equal or above 50 (Mendling et al., 2010).

<sup>4</sup><https://nodered.org/docs/user-guide/nodes#switch>

<sup>5</sup><https://nodered.org/blog/2016/06/14/version-0-14-released>

<sup>6</sup><https://nodered.org/docs/user-guide/editor/workspace/subflows>

### 2.4 Layout

**Wiring Style Consistency (WSC).** The wires connecting the nodes should follow a consistent wiring style, to differentiate main/correct scenarios from exceptional/wrong ones (Reggio et al., 2011; Reggio et al., 2012; Ambler, 2005; Unhelkar, 2005). Since Node-RED flows may handle several scenarios, as it happens for classic programming languages concerning conditional statements, different wiring styles may be adopted within the same flow. For example, a “straight, from left to right, top-down” style to wire all the nodes participating in a correct scenario, and a “cascade” style to wire all the nodes participating in a wrong scenario.

**Wiring Style Tidiness (WST).** The wires connecting the nodes should be long enough to clearly show the starting/ending nodes and avoid any overlapping, whether possible (Ambler, 2005). Wires should be drawn in the order they enter/leave a node. The node joining multiple wires should be placed at the level of the node where such wires originated.

## 3 SELECTED NODE-RED SYSTEMS

To conduct the experiment, later discussed in Section 4, we selected two existing Node-RED systems that were developed by former students of the master course Data Science and Engineering (Genova, Italy), as part of the last year project.

The systems, named **DiaMH** and **WikiDataQuerying**, present some common Node-RED comprehensibility issues derived from an undisciplined and basic Node-RED usage (i.e., the teacher of that course was not involved in our research and thus students developed the systems without following any guideline).

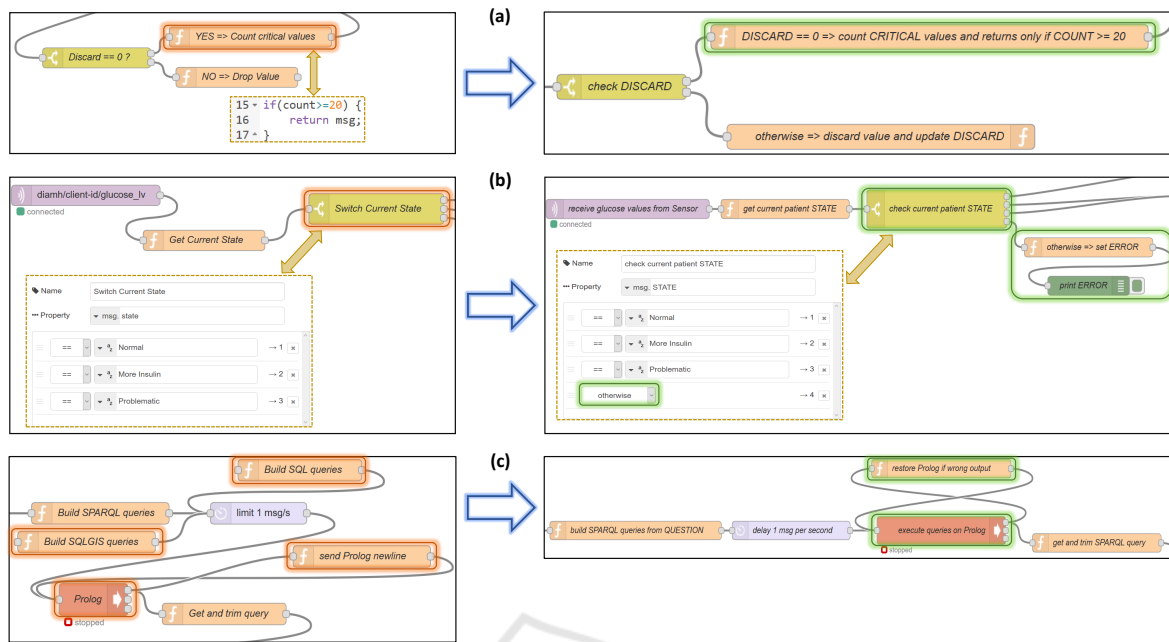


Figure 1: Issues in Node-RED Systems and Guidelines Application.

### 3.1 DiaMH System

DiaMH is a simulated Diabetes Mobile Health IoT system which monitors a diabetic patient by collecting glucose values using a wearable sensor, sends notifications to the patient’s smartphone about the monitored data, and, based on some logical computations involving a cloud-based healthcare system and realistic data patterns, determines the patient’s health state (i.e., Normal, More Insulin required, or Problematic) and, when needed, orders insulin injections to a wearable insulin pump. It consists of 71 nodes and 63 wires.

### 3.2 WikiDataQuerying System

WikiDataQuerying is a web query service used to select textual geospatial questions from a predefined list shown in an HTML page, and query WikiData<sup>7</sup> knowledge base, by first restructuring the selected questions into SPARQL query language and formatting them using a Prolog grammar. The results of the queries can be triples adhering to the Resource Description Framework (RDF) language or boolean answers. It consists of 25 nodes and 27 wires.

### 3.3 Applying Guidelines to Node-RED Systems

For the selected Node-RED systems, only some of the aforementioned guidelines have to be applied. However, even such simple systems can hide several comprehensibility issues. Despite their simplicity, involving mainly core Node-RED nodes, DiaMH works in the thorny context of the healthcare and WikiDataQuerying must provide prompted feedback to the user’s requests. Therefore, producing Node-RED flows that adhere to our proposed guidelines may improve the comprehensibility level during flows inspection and development, and facilitate the subsequent engineering stages, such as maintainability and testing.

Just few of the issues we found from a high-level flows analysis of the systems are shown in Figure 1 and discussed in the following.

Most of the nodes names do not clarify their behaviours, forcing the developer to inspect the nodes contents and settings in order to comprehend them. Therefore, **NNB** can be applied to clarify the nodes purposes, by making explicit in their names the performed actions and the used variables in uppercase (see nodes names in Figure 1, changed from left to right).

Sub-flow **(a, left)** of Figure 1 presents the function node (i.e., a core Node-RED node used to implement customized JavaScript functions<sup>8</sup>) named *Yes*

<sup>7</sup><https://www.wikidata.org>

<sup>8</sup><https://nodered.org/docs/user-guide/nodes#function>



=> *Count critical values*, which performs several actions and then sends a message to subsequent nodes (not shown in Figure 1) only when a certain condition holds (i.e., `if count >= 20`, see lines 15-17 in Figure 1 left associated with the node); from an unaware Node-RED developer perspective, this may unexpectedly block the execution of the sub-flow until the condition is satisfied, even if the nodes are graphically connected by means of wires. This comprehensibility issue emerged in several topics posted on the main Node-RED forum<sup>9</sup> and can be solved by applying **NNB**, as previously mentioned, by renaming the function node as, e.g., `DISCARD == 0 => count CRITICAL values and returns only if COUNT >= 20`, in order to make explicit the condition it satisfies from the preceding switch node (i.e., `DISCARD == 0`), the core behaviour (i.e., `count CRITICAL values`), and the hidden condition for the message to return (i.e., `returns only if COUNT >= 20`). Sub-flow (**a, right**) of Figure 1 is the result.

In sub-flow (**b, left**) of Figure 1, the switch node *Switch Current State* hides a severe issue: it considers only three possible values for the `msg.state` variable, but the check will fail and idle the sub-flow execution if any unexpected event sets the variable to a different value before the switch node. This problem was solved in the dawn era of Node-RED<sup>10</sup> by introducing an “otherwise” entry to handle all the alternative conditions, but the average Node-RED developer may still miss to use it in favour of a more explicit, but incomplete set of conditions<sup>11</sup>. By applying **CCC**, the “otherwise” condition is added to the switch node (see the change in the configuration panel from left to right in Figure 1), while the application of **WSC** displays in a cascaded wiring style the newly introduced exceptional scenario. Sub-flow (**b, right**) of Figure 1 is the result.

Sub-flow (**c, left**) of Figure 1 presents several issues. First, the wires do not follow any consistent wiring style (e.g., build the flow by wiring nodes from top to bottom or from left to right, avoiding crossing wires), which reduces the overall comprehensibility<sup>12</sup>. Second, a loop is generated between *Prolog* and *send Prolog newline* nodes; although in this scenario finding the loop is rather simple, it may be harder to detect and produce a weird outcome or overheat the CPU, when

more wires are involved<sup>13</sup>. Third, the sub-flow shows three function nodes connected through output pins to *limit 1 msg/s* node, but only *Build SPARQL queries* actually contributes to the flow, since the other two nodes have no entering wires, making them inactive; this case in particular is easy to detect, but could be hard to understand for a novel Node-RED developer, in case she forgets to trace a wire between two nodes to specify the input source or the output destination of a node, without receiving any explicit warning from the Node-RED environment. This issue often arises when there is a need for debugging a flow and some nodes have to be temporary disconnected from it<sup>14</sup>. By applying **WSC** and **WST**, a more consistent and tidier wiring style is generated, to highlight the loop and avoid further entangles, while **NEC** is used to remove the miracle nodes originally named *Build SQL queries* and *Build SQLGIS queries* (i.e., those having input pins but no entering wires, in fact inactive). Sub-flow (**c, right**) of Figure 1 is the result.

## 4 EXPERIMENTAL EVALUATION

Based on the Goal Question Metric (GQM) template (Van Solingen et al., 2002), the main goal of our experiment can be defined as follows: “Evaluate the effect of the guidelines in Node-RED flows comprehension”, with the purpose of understanding if the guidelines are able to improve the *comprehension level* of Node-RED flows and the *time* required to complete tasks pertaining such flows; therefore, consequently, the overall *efficiency* is computed as:  $comprehension\ level \div time$ .

The *perspective* is of: a) Node-RED developers, using it for their own purpose and/or sharing artefacts with the community, who may be interested to consider a disciplined technique to develop Node-RED flows using our guidelines; b) teachers and instructors interested to offer courses and tutorials on Node-RED and c), researchers interested in focusing their research activities and study improvements or constraints to the Node-RED language.

Thus, our research questions are:

**RQ1.** Does the comprehension level of Node-RED flows vary when our guidelines are applied?

**RQ2.** Does the comprehension time of Node-RED flows vary when our guidelines are applied?

<sup>9</sup><https://discourse.nodered.org/t/function-node-stopping/7017>

<sup>10</sup><https://github.com/node-red/node-red/issues/88>

<sup>11</sup><https://discourse.nodered.org/t/switch-node-not-consistent/11908>

<sup>12</sup><https://discourse.nodered.org/t/help-simplifying-flow/8765>

<sup>13</sup><https://discourse.nodered.org/t/cpu-hogging-to-100/2944>

<sup>14</sup><https://discourse.nodered.org/t/how-to-comment-out-a-node/1106>

**RQ3.** Does the efficiency of completing tasks pertaining Node-RED flows vary when our guidelines are applied?

To quantitatively investigate the research questions, we used ad-hoc questionnaires containing 16 comprehension questions for each experimental object. We measured the comprehension level of Node-RED flows as the number of correct answers on the total, the time required to provide such answers, and the efficiency as the ratio between the comprehension level and the time required to provide such answers (i.e., the number of correct answers divided by the time is a proxy for measuring the efficiency construct).

Table 2 summarizes the main elements of the experiment, following the guidelines by Wohlin *et al.* (Wohlin *et al.*, 2012).

Table 2: Overview of the Experiment.

<b>Goal</b>	Evaluate the effect of the guidelines in Node-RED flows comprehension
<b>Quality focus</b>	Pertaining Node-RED tasks, we evaluate: (i) Comprehension (ii) Time (iii) Efficiency
<b>Context</b>	<b>Objects:</b> DiaMH and WikiDataQuerying Node-RED systems <b>Participants:</b> 10 Computer Science master students
<b>Null Hypotheses</b>	(i) No effect on comprehension (ii) No effect on time (iii) No effect on efficiency
<b>Treatments</b>	Non-compliant (−) and Compliant (+) Node-RED flows
<b>Dependent variables</b>	(i) TotalComprehension to complete Node-RED tasks (ii) TotalTime to complete Node-RED tasks (iii) TotalEfficiency to complete Node-RED tasks

In the following, we describe in detail: treatments, objects, participants, experiment design, hypotheses, variables, procedure, and other aspects of the experiment.

#### 4.1 Treatments

Our experiment has one independent variable (main factor) and two treatments: *Non-compliant* and *Compliant* Node-RED flows. Non-compliant Node-RED flows (in the following, characterized by symbol −) are those produced without following our guidelines, while compliant Node-RED flows (in the following, characterized by symbol +) are those produced following our guidelines.

#### 4.2 Objects

The objects of the study are DiaMH and WikiDataQuerying systems, presented in Section 3. Both were developed by former students of another course. We limited each object to just a comparable (in size and complexity) flow of the original behaviour, consisting of 20 nodes and 23 wires for DiaMH and 21 nodes and 21 wires for WikiDataQuerying, employing mostly Node-RED core nodes.

We carefully inspected and tested both the flows of the systems; these two flows correspond to the non-compliant treatment (−), since our guidelines were not adopted during their implementations. Then, an author of the paper applied the guidelines discussed in Section 2 to the initial flows, producing two equivalent compliant versions (+), while another author double checked the newly produced flows. In total, we used four Node-RED flows for executing the experiment: DiaMH−, DiaMH+, WikiDataQuerying−, WikiDataQuerying+.

#### 4.3 Participants

We involved ten Computer Science master students of the University of Genova (Italy), that were attending a course on advanced software engineering. The total number of students enrolled in the course was 15, which is basically the average number of students enrolled in any Computer Science Master Course in Genova.

They had average knowledge of Software Engineering, UML and JavaScript (the Node-RED core programming language), and few experience in Node-RED and flow-based programming, that was provided in another course related to Node-RED development.

#### 4.4 Experiment Design

Before the experiment, all the participants were involved in a 4-hours lecture split in two days about Node-RED theory and practice using the tool. Participants were provided with material to understand the main Node-RED core nodes, samples of flows and sub-flows to reproduce/change, and questions to answers about comprehensibility issues of the flows similar to those we asked for the later experiment. Participants were not informed about the guidelines, and therefore, about the treatments.

Due to the limited number of participants (only ten), we adopted a counterbalanced experiment design ensuring each participant to work in two tasks on the two different objects, receiving each time a different treatment. Since participants had the same experience

in Node-RED, acquired by attending another course, we randomly split them into four groups (see Table 3), balancing the representatives for each group. Each participant had to work first on **Task 1** on an object with a treatment, then in **Task 2** on the other object with the other treatment.

#### 4.5 Dependent Variables and Hypotheses Formulation

Our experiment had three dependent variables, on which the treatments were compared measuring three different constructs to answer our three research questions: (a) *Comprehension* of the Node-RED flows (measured by variable *TotalComprehension*), (b) *Time* required to answer the questions pertaining the Node-RED flows (measured by variable *TotalTime*), (c) *Efficiency* in completing the tasks pertaining the Node-RED flows (measured by variable *TotalEfficiency*).

For each treatment:

- *TotalComprehension* was computed by summing up the number of correct answers of each participants;
- *TotalTime* was computed as the difference between the stop time of the last question and the start time of the first question, where timing was tracked down in the time sheet by each participant;
- *TotalEfficiency* was derived by the two previously computed variables, as:

$$TotalEfficiency = \frac{TotalComprehension}{TotalTime}$$

Since we could not find any previous empirical evidence pointing out a clear advantage of one treatment versus the other, we formulated the following three null hypotheses as non-directional, with the objective to reject them in favour of alternative ones:

- $H_{0a}: TotalComprehension^- = TotalComprehension^+$
- $H_{0b}: TotalTime^- = TotalTime^+$
- $H_{0c}: TotalEfficiency^- = TotalEfficiency^+$

#### 4.6 Material, Procedure and Execution

To estimate the comprehensibility of the tasks to provide to the participants and the time required to complete them, we conducted a pilot experiment with three participants: two master students in Computer Science not involved in the experiment and one of the authors of this paper. On average, the time required to complete both tasks was about 105 minutes, with 5 errors. Given such results, we tried to remove any ambiguity from the questions.

Then, we uploaded the material on the Moodle module of the course from which the participants were selected, consisting, for each group of Table 3, of: two Node-RED flows (one per system/treatment), two questionnaires containing 16 questions each, and a post-questionnaire to fill after the completion of the two questionnaires containing seven further questions.

Each questionnaire presented exactly 7 open questions and 9 multiple choices questions, in order to keep the perceived complexity of both tasks as equivalent as possible. Questions ranged from comprehending the general behaviour of the provided Node-RED flows, like identifying the names and the number of nodes involved in certain activities, detecting the presence of loops and missing conditions in switch nodes, counting the number of intersections among wires, to listing some simple maintenance tasks to do on the flows. Concerning multiple choices questions, only one answer among the proposed was correct and counted 1 point each, while for open questions we gave 1 point to totally correct answers and 0 otherwise. For each object (i.e., DiaMH and WikiDataQuerying), the questions asked to the participants were exactly the same, independently from the treatment that had occurred (i.e., non-compliant or compliant).

The participants had to complete each task in the order defined by the group they were assigned to, and to stop each task only when completed. For each task, participants had to import the corresponding Node-RED flow into Node-RED and, for each question, track start time, answer the question, and track stop time.

Finally, the participants were asked to complete the post-experiment questionnaire, to collect insights about their skills and motivations for the obtained results. Questions were about the perceived complexity of the two tasks, the exercise usefulness, the feelings and the preferences between the styles of the two flows, and the competencies required to complete the tasks. Answers were provided on a Likert scale ranging from one (Strongly Agree) to five (Strongly Disagree).

#### 4.7 Analysis

Because of the sample size and mostly non-normality of the data (measured with the Shapiro–Wilk test (Shapiro and Wilk, 1965)), we adopted non-parametric test to check the three null hypotheses.

Since participants answered to the questions on the two different objects (DiaMH and WikiDataQuerying) with the two possible treatments (non-compliant and compliant), we used a paired Wilcoxon test to compare the effects of the two treatments on each participant.

To measure the magnitude of the effects of the two treatments, we used the non-parametric Cliff's delta

Table 3: Experimental Design (+ for Compliant treatment, - for Non-Compliant treatment).

	Group A	Group B	Group C	Group D
Task 1	DiaMH <sup>+</sup>	DiaMH <sup>-</sup>	WikiDataQuerying <sup>+</sup>	WikiDataQuerying <sup>-</sup>
Task 2	WikiDataQuerying <sup>-</sup>	WikiDataQuerying <sup>+</sup>	DiaMH <sup>-</sup>	DiaMH <sup>+</sup>

(d) effect size (Grissom and Kim, 2005), which is considered small (S) for  $0.148 \leq |d| < 0.33$ , medium (M) for  $0.33 \leq |d| < 0.474$ , and large (L) for  $|d| \geq 0.474$ .

We decided to accept the customary probability of 5% of committing Type-I-error (Wohlin et al., 2012), i.e., rejecting the null hypothesis when it is actually true.

## 5 RESULTS

In this section, the effect of the main factor on the dependent variables (*TotalComprehension*, *TotalTime*, and *TotalEfficiency*), as resulted from the experiment, and the post-experiment questionnaires are discussed.

Table 4 summarizes the essential *Comprehension*, *Time*, and *Efficiency* descriptive statistics (i.e., median, mean, and standard deviation) per treatment, and the results of the paired Wilcoxon analysis conducted on the data from the experiment with respect to the three dependent variables.

### 5.1 $H_{0a}$ : Comprehension (RQ1)

Fig. 2 summarizes the distribution of *TotalComprehension* by means of boxplots. Observations are grouped by treatment (non-compliant or compliant). The y-axis represents the average comprehension measured as number of correct answers on the 16 questions for each treatment, where score = 16 represents the maximum value of comprehension and corresponds to provide correct answers to all the 16 questions.

The boxplots show that the participants achieved a better comprehension level when working on the compliant Node-RED flows (median 13.5) with respect to those working on non-compliant flows (median 9.5).

By applying a Wilcoxon test (paired analysis), we found that the difference in terms of comprehension is statistically significant, as testified by p-value = 0.00903. Therefore, we can reject the null hypothesis  $H_{0a}$ . The effect size is large ( $d = -0.69$ ).

**To Answer RQ1:** The adoption of the guidelines significantly improves the level of comprehension of the Node-RED flows.

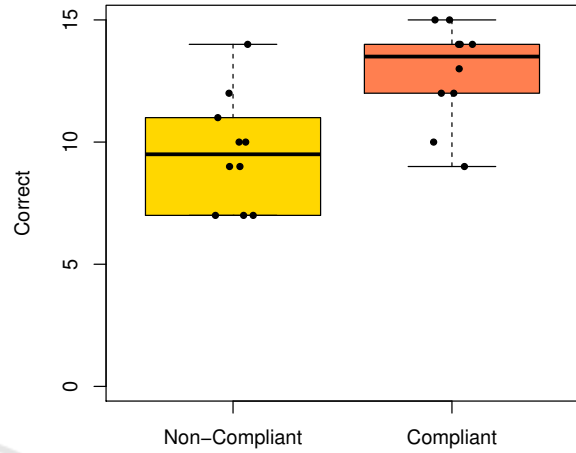


Figure 2: Boxplots of Comprehension.

### 5.2 $H_{0b}$ : Time (RQ2)

Fig. 3 summarizes the distribution of *TotalTime* by means of boxplots, where the y-axis represents the total time to answer the 16 questions for each treatment. The boxplots show that the participants needed slightly more time to answer the questions pertaining the objects with the non-compliant treatment w.r.t. those answering the questions pertaining the objects with the compliant treatment (58.5 versus 57.0 minutes respectively in the median case).

By applying a Wilcoxon test (paired analysis), we found that the overall difference is marginally significant (p-value = 0.04883). Therefore, we can reject the null hypothesis  $H_{0b}$ . The effect size is small ( $d = 0.17$ ).

**To Answer RQ2:** The adoption of the guidelines marginally reduces the time required to answers the questions pertaining the Node-RED flows.

### 5.3 $H_{0c}$ : Efficiency (RQ3)

Fig. 4 summarizes the distribution of *TotalEfficiency* by means of boxplots.

The boxplots show that participants working on the objects with the compliant treatment outperformed in terms of efficiency those working with the objects with the non-compliant treatment (medians 0.254 versus 0.180, respectively).



Table 4: Descriptive statistics per treatment and results of paired Wilcoxon test.

Dependent Variable	Non-Compliant Treatment (⁻)			Compliant Treatment (+)			p-value	Cliff's Delta
	Median	Mean	St. Dev.	Median	Mean	St. Dev.		
TotalComprehension	9.500	9.600	2.319	13.500	12.800	2.044	0.00903	-0.69 (L)
TotalTime	58.500	71.100	38.484	57.000	59.100	25.291	0.04883	0.17 (S)
TotalEfficiency	0.180	0.179	0.109	0.254	0.253	0.100	0.00586	-0.36 (M)

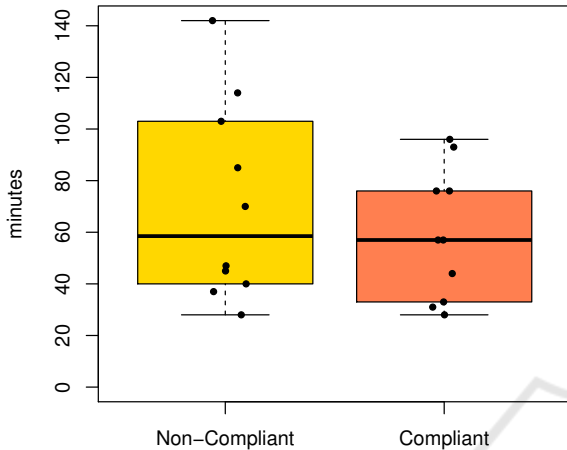


Figure 3: Boxplots of Time.

By applying a Wilcoxon test (paired analysis), we found that the overall difference is statistically significant, as shown by the p-value (p-value = 0.00586). Therefore, we can reject the null hypothesis  $H_{0c}$ . The effect size is medium ( $d = -0.36$ ).

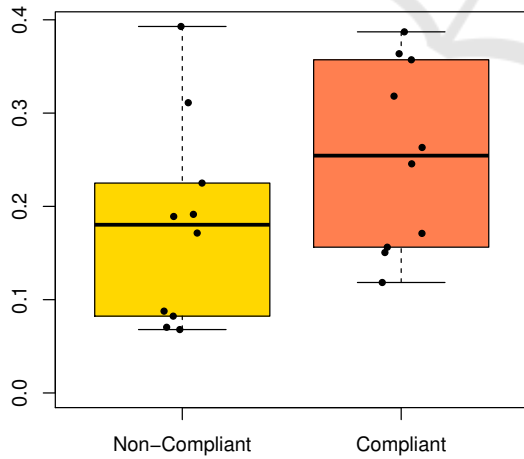


Figure 4: Boxplots of Efficiency.

**To Answer RQ3:** The adoption of the guidelines increases the overall efficiency in the comprehension of the Node-RED flows.

### 5.4 Post Experiment

When participants had to fill the post-experiment questionnaire, they were unaware of the guidelines and, therefore, of the two treatments. For this reason, the actual questions were formulated as a comparison between the flows they had worked on in the two tasks, keeping track of which treatment occurred on them, according to the group the participants were assigned to. Thus, for instance, question **PQ1** was originally formulated as *Comprehending the Node-RED flow in Task 1 was harder than the Node-RED flow in Task 2*. In Table 5 the post-experiment questionnaire has been adjusted in order to clarify the purpose of our experiment. Table 5 reports also the medians of the answers given by the participants. The possible choices for each answer, on a 5-point Likert scale, were: Strongly Agree, Agree, Unsure, Disagree, Strongly Disagree.

Table 5: Adjusted post-experiment questionnaire.

ID	Question	Median
PQ1	Comprehending the non-compliant Node-RED flow was harder than the compliant one	Unsure
PQ2	In your opinion, developing the non-compliant Node-RED flow is harder than the compliant one	Agree
PQ3	In your opinion, maintaining the non-compliant Node-RED flow is harder than the compliant one	Agree
PQ4	The names of the nodes and the variables in the non-compliant Node-RED flow were less useful for the comprehension than in the compliant one	Unsure
PQ5	The wiring style to connect nodes in the non-compliant Node-RED flow was less useful for the comprehension than in the compliant one	Agree
PQ6	I found the exercise useful	Agree
PQ7	I had enough knowledge to answer the questions	Agree

As the Table 5 shows, participants did not perceive any difference in the complexity while trying to comprehend the Node-RED flows using each treatment (**PQ1**), but believed that developing and maintaining such flows may result more complex with the non-compliant treatment (**PQ2-3**). The names of the nodes

and of the used variables in the two treatments had no significant impact in the overall comprehension (PQ4), whereas the wiring style was better perceived in the case of the compliant treatment (PQ5). In general, participants found the exercise useful (PQ6) to the course of their studies, and fitting their knowledge in Node-RED (PQ7), in part acquired by attending the 4-hours lecture preceding the experiment.

## 5.5 Discussion

Given the results of the experiment, all null hypotheses can be rejected. The guidelines are generally beneficial to the comprehension level and reduce the time required to complete Node-RED tasks. Consequently, the overall efficiency is also positively affected.

One of the main reasons of success of the guidelines was producing flows that follow a consistent and tidy wiring style, by means of WSC and WST, which improved the capability of detecting loops and reduced entangles among wires. This is corroborated by PQ5 of Table 5. In fact, the questions pertaining comprehensibility issues about wires and loops presented generally better outcomes for the flows compliant with our guidelines. For example, a question in both questionnaires requires identifying the number of intersections between wires. While we had only 1 error for the flows compliant with our guidelines, for the non-compliant flows the errors amounted to 8.

On the other hand, by the feeling of the participants (PQ4), giving proper names to nodes was not so relevant for the comprehension. This is contradicted by the results of the experiment, since the importance of names, given by NNB guideline, resulted to be helpful in indirectly answering several questions on both systems. For instance, we had two open questions specifically asking the names of the nodes responsible for a certain behaviour (e.g., *Which node (provide name) displays on a web page the WikiData answer to the user's question?*), which resulted in a total of 4 errors for the flows compliant with our guidelines against 10 for the non-compliant ones. We had also two questions in both questionnaires asking about which data was changed/returned after the completion of a certain activity (e.g., *Which data are saved just after the HTTP request to URL?*): while we had just 1 error in the flows compliant with our guidelines, 5 errors were counted for the non-compliant cases. One question asked about the effective contributions of the nodes in a selected portion of the flows (i.e., if they were able to transform the message they had received); by using NEC guideline, the inactive nodes (e.g., those added for debugging purposes) were removed from the flows compliant with the guidelines, but not from the non-

compliant flows. In this case, while participants easily identified the contributions of the remaining nodes, they failed (4 errors) in identifying the inactive ones in the non-compliant flows. Finally, there was a question asking to list all the files in the file system used by the flows, which resulted in 1 error for the flows compliant with our guidelines against 4 for the non-compliant ones. Indeed, by following our guidelines, the nodes names were formulated to make their behaviours more explicit, as well as the main used variables, hence reducing the overall errors in the comprehension, as summarized by the statistics data of Table 4.

From PQ1, participants did not have a clear opinion on which treatment was easier to comprehend, while they agreed that flows produced without following our guidelines would be harder to develop and maintain (PQ2-3). Concerning comprehensibility complexity, we speculate that the uncertainty of the participants is due to the domain of the two systems: while DiaMH presents the MQTT node (i.e., a core Node-RED node used to establish a communication from/to entities and flows using the MQTT protocol<sup>15</sup>) as the most complex node, WikiDataQuerying refers to WikiData repository and to Prolog and SPARQL languages, which could deviate from their average academic background. Finally, the participants recognized that the tasks they completed did not require excessive knowledge of Node-RED and were helpful for their current/next academic studies (PQ6-7).

To conclude, the proposed guidelines resulted useful in terms of comprehension level, time, and overall efficiency. This may suggest Node-RED developers to apply them to reduce the comprehensibility issues of the flows they will produce, deploy and share. At the same time, the guidelines may turn useful to the designers of the Node-RED language, who may want to fix some of the issues exposed in the paper, by introducing additional features in future Node-RED releases. Just to mention few possible additions: (i) nodes resizing in height and width, to highlight the most important nodes and make long names more readable, (ii) general warnings, to notify the presence of unused variables, incomplete conditions within switch nodes, and loops, and (iii) jumps between wires, to graphically handle colliding wires.

## 5.6 Threats to Validity

The threats to validity that could affect our experimentation are: internal, construct, conclusion and external (Wohlin et al., 2012).

*Internal validity threats:* these threats concern factors which may affect the dependent variables. The

<sup>15</sup><https://cookbook.nodered.org/mqtt/>

participants had to complete two tasks; therefore, a fatigue/learning effect may have intervened. However, since they had a break between the two tasks and they previously completed some exercises about Node-RED comprehensibility issues, we expect this effect to be limited. Another threat is the subjectivity in the objects selection. The objects were flows chosen from a list of systems developed by former master students of another course related to Node-RED, and were comparable in size and complexity and composed of mostly Node-RED core nodes.

*Construct validity threats:* these threats concern how comprehension and time were measured. The correctness of the answers was checked by one of the authors, who also measured the execution time, based on the time sheets filled by the participants. The statistics data (i.e., median, mean, and standard deviation) and the results of the paired Wilcoxon analysis were computed using Excel and R<sup>16</sup>.

*Conclusion validity threats:* these threats concern the limited sample size of the experiment (ten master students), which may have affected the statistical tests. Unfortunately, this is the average number of students of any Computer Science Master Course in Genova, so it is difficult for us to conduct experiments with more participants.

*External validity threats:* these threats can limit the generalization of the results and, in our case, concern the use of students as experimental participants. Our participants had few knowledge of Node-RED, therefore more expert Node-RED developers may produce a different outcome. We intend to replicate our experiment with more complex systems and more expert developers.

## 6 RELATED WORK

As already mentioned in Section 2, our guidelines have been inspired by several works on UML and BPMN concerning the quality of the produced models, while none specifically treat guidelines for developing comprehensible Node-RED flows.

The guidance provided to Node-RED developers in implementing nodes is limited to a simple set of principles<sup>17</sup>, like nodes should be “simple to use” and “consistent” in their behaviour, and few unofficial design patterns<sup>18</sup> to make flows easier to understand and reuse.

<sup>16</sup><https://www.r-project.org/>

<sup>17</sup><https://nodered.org/docs/creating-nodes>

<sup>18</sup><https://medium.com/node-red/node-red-design-patterns-893331422f42>

In a recent industrial work (Bröring et al., 2019), Bröring et al. propose an approach to automatically collect metadata from Node-RED flows and nodes, and feed a knowledge base for future analyses, such as nodes quality ratings, downloads data, and nodes dependencies. Although that work considers several quality aspects of Node-RED, like selecting the most suitable solutions to integrate within a system, it does not provide to users any development guideline.

Prehofer and Chiarabini (Prehofer and Chiarabini, 2015) identify the differences between mash-up tools for IoT systems, like Node-RED, and model-based approaches for the IoT, and propose an approach to exploit both their benefits: the simplicity of mash-up tools in systems development and the strengths of models to formalize a behaviour and have it checked by a model checker. However, in that paper, the quality checks of IoT systems are not oriented to the comprehensibility issues that may emerge during flows development using mash-up tools.

Mendling et al. (Mendling et al., 2010) provide 7 guidelines, built on empirical insights, as a response to the lack of practical solutions to improve the quality of business process models. Some of these guidelines have been adapted for our work. For example, “Use verb-object activity label” (G6), to reduce the ambiguity of the constructs in a model, particularly useful for the large number of nodes collaborating within Node-RED flows, as well as “Use as few elements in the model as possible” (G1) and “Decompose a model with more than 50 elements” (G7), to reduce flows complexity.

Unhelkar, in his book (Unhelkar, 2005), focuses on syntax, semantics and aesthetic checks of UML 2.0 diagrams. Although UML is quite different from Node-RED in many aspects, since it operates at a design stage and involves constructs that are hardly comparable to Node-RED nodes and wires, in the book there are some aesthetic checks concerning activity diagrams that we have embodied in our work. In particular, it is important to adopt a consistent style to differentiate regular from exceptional scenarios and to balance overpopulated diagrams by redistributing the included constructs.

Reggio et al. (Reggio et al., 2012; Reggio et al., 2011) face the problem of quality in business process modelling. They propose an empirical method for helping the modeller in choosing among five business process modelling styles, that differ in terms of abstraction and precision. For instance, a more precise style requires each construct to declare all the participants, the objects and the used data in capital letters, like we did in our work, to make explicit the data used by each node.

Ambler proposes (Ambler, 2005) several guidelines addressing both general and UML-specific modelling issues, with the aim of improving the effectiveness of the produced models. Some of these guidelines can be applied even in Node-RED, since they use very general terms and descriptions. For what concerns UML, we focused on activity diagrams since they represent sequences of actions similarly to Node-RED flows. The guidelines suggest, for instance, to avoid black-hole and miracle nodes (i.e., nodes without a leaving/entering line), that may indicate a missing interaction, and to check that the guards within decision points are always complete.

## 7 CONCLUSION AND FUTURE WORK

In this paper, we have proposed a preliminary set of guidelines, addressing some common Node-RED comprehensibility issues, which may help the Node-RED developers in producing flows that are easier to comprehend and suitable for future maintenance and testing activities. To evaluate the guidelines, we have conducted an experiment involving ten master students.

The results have shown that the adoption of the guidelines increased the overall efficiency in Node-RED flows comprehension, by reducing the number of errors and the time required to complete comprehension tasks over some provided Node-RED flows. In addition to supporting the Node-RED developers in flows development, the proposed guidelines pinpoint some Node-RED comprehensibility issues that might be solved in future releases of the tool, by adding features like: (i) nodes resizing in height and width, to highlight the most important nodes and make long names more readable, (ii) general warnings, to notify e.g., the presence of unused variables, incomplete conditions within switch nodes, and loops, and (iii) graphical jumps between wires, to handle colliding wires.

The proposed guidelines are just a preliminary set of those that we are planning to address further Node-RED comprehensibility issues that did not emerge from the Node-RED systems we chose; for instance, how to avoid loops by transforming graph-based flows into tree-based ones. As future work, we intend to involve a larger participants pool in the experimental evaluation of the guidelines, including Node-RED designers, and to implement a checker tool to automatically detect the comprehensibility issues from the Node-RED flows failing our guidelines, and fix them accordingly.

## REFERENCES

- Ambler, S. W. (2005). *The elements of UML (TM) 2.0 style*. Cambridge University Press.
- Bröring, A., Charpenay, V., Anicic, D., and Püech, S. (2019). Nova: A knowledge base for the Node-RED IoT ecosystem. In *Proceedings of ESWC 2019*, pages 257–261. Springer.
- Clerissi, D., Leotta, M., Reggio, G., and Ricca, F. (2018). Towards an approach for developing and testing node-red iot systems. In *Proceedings of EnSEmble@ESEC/SIGSOFT 2018*, pages 1–8.
- Grissom, R. J. and Kim, J. J. (2005). *Effect sizes for research: A broad practical approach*. Lawrence Erlbaum Associates Publishers.
- Lange, C. F., DuBois, B., Chaudron, M. R., and Demeyer, S. (2006). An experimental investigation of UML modeling conventions. In *Proceedings of MODELS 2006*, pages 27–41. Springer.
- Leotta, M., Clerissi, D., Olianas, D., Ricca, F., Ancona, D., Delzanno, G., Franceschini, L., and Ribaud, M. (2018). An acceptance testing approach for internet of things systems. *IET Software*, 12(5):430–436.
- Martin, R. C. (2003). *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR.
- Mendling, J., Reijers, H. A., and van der Aalst, W. M. (2010). Seven process modeling guidelines (TPMG). *Information and Software Technology*, 52(2):127–136.
- Morrison, J. P. (2010). *Flow-Based Programming: A new approach to application development*. CreateSpace.
- Prehofer, C. and Chiarabini, L. (2015). From Internet of Things mashups to model-based development. In *Proceedings of COMPSAC 2015*, volume 3, pages 499–504. IEEE.
- Reggio, G., Leotta, M., and Ricca, F. (2011). "Precise is better than light" A document analysis study about quality of business process models. In *Proceedings of EmpiRE 2011*, pages 61–68. IEEE.
- Reggio, G., Leotta, M., and Ricca, F. (2014). Who knows/uses what of the UML: A personal opinion survey. In *Proceedings of MODELS 2014*, volume 8767 of LNCIS, pages 149–165. Springer.
- Reggio, G., Leotta, M., Ricca, F., and Astesiano, E. (2012). Business process modelling: Five styles and a method to choose the most suitable one. In *Proceedings of EESSMod@MODELS 2012*, pages 8:1–8:6. ACM.
- Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.
- Unhelkar, B. (2005). *Verification and Validation for Quality of UML 2.0 Models*, volume 42. John Wiley & Sons.
- Van Solingen, R., Basili, V., Caldiera, G., and Rombach, H. D. (2002). Goal Question Metric (GQM) approach. *Encyclopedia of software engineering*.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.