JOT
JOURNAL OF
OBJECT TECHNOLOGY

Journal of Object Technology | **RESEARCH ARTICLE**

Simulink bus usage in practice: an empirical study

Tiago Amorim[†,1], Alexander Boll[†,2], Ferry Bachman[3], Timo Kehrer[2], Andreas Vogelsang[1], and Hartmut Pohlheim[3]
[†]These authors contributed equally to this work and share first authorship.
[1]University of Cologne, Germany
[2]University of Bern, Switzerland
[3]Model Engineering Solutions, Germany

ABSTRACT Matlab/Simulink is a graphical modeling environment that has become the de facto standard for the industrial model-based development of embedded systems. Practitioners employ different structuring mechanisms to manage Simulink models' growing size and complexity. One important architectural element is the so-called bus, which can combine multiple signals into composite ones, thus, reducing a model's visual complexity. However, when and how to effectively use buses is a non-trivial design problem with several trade-offs. To date, only little guidance exists, often applied in an ad-hoc and subjective manner, leading to suboptimal designs. Using an inductive-deductive research approach, we conducted an exploratory survey among Simulink practitioners and extracted bus usage information from a corpus comprising 433 open-source Simulink models. We elicited 22 hypotheses on bus usage advantages, disadvantages, and best practices from the data, whose validity was later tested through a confirmatory survey. Our findings serve as requirements for static analysis tools and pave the way toward guidelines on bus usage in Simulink.

KEYWORDS Modeling, Simulink, Buses, Empirical study.

1. Introduction

Over the last two decades, Matlab/Simulink (Simulink, for short) has become the *de facto* standard for the industrial model-based development of embedded systems in various domains (e.g., automotive, avionics, industrial automation, medicine) (Liggesmeyer & Trapp 2009; Vanherpen et al. 2015), with millions of users (Popoola & Gray 2021) and thousands of companies[1] employing it. While Simulink started as a tool mainly for modeling and simulating single controllers, today, it can describe large systems comprising thousands of blocks communicating over signals (Boll et al. 2021). This increasing complexity has triggered research on properly structuring large-scale Simulink models by adopting well-known software design principles (Dajsuren et al. 2013; Dajsuren 2015; Jaskolka, Scott, et al. 2020; Jaskolka, Pantelic, et al. 2020; Pantelic et al. 2017; Whalen et al. 2014). Current studies revolve around hierarchical decomposition, encapsulation, and information hiding, aiming at classical quality attributes such as modularity, coupling, and cohesion.

The *subsystem* and the *bus* are Simulink's architectural elements that address some of the aforementioned properties. The former provides the capability for encapsulating blocks and other nested subsystems. The latter allows combining individual signals into composite signals to keep them organized and reduce visual clutter[2]. However, buses have gained considerably less attention in the literature than subsystems, despite their importance for large and complex models. For instance, in one of the largest datasets (Boll et al. 2023) of public Simulink projects (similar to those used in (Boll et al. 2021; Shrestha et al. 2022)), we found models that do use buses have a median of 22 subsystems and 239 blocks. In comparison, models without buses feature three subsystems and 29 blocks in the median.

Additionally, guidelines and best practices on bus use are

JOT reference format:
Tiago Amorim, Alexander Boll, Ferry Bachman, Timo Kehrer, Andreas Vogelsang, and Hartmut Pohlheim. *Simulink bus usage in practice: an empirical study*. Journal of Object Technology. Vol. 22, No. 2, 2023.

http://dx.doi.org/10.5381/jot.2023.22.2.a12

[1] https://enlyft.com/tech/products/simulink

[2] https://www.mathworks.com/help/simulink/ug/composite-signal-techniques.html

severely limited. The existing ones, namely the MathWorks Advisory Board (MAB) guidelines and the Motor Industry Software Reliability Association (MISRA) guidelines (MathWorks Advisory Board 2020; Misra 2023), provide little guidance on bus usage, apart from naming advice, label position, and suggesting not to mix MUX and bus elements. However, more advanced bus usage principles are implicit, ad-hoc, and subjective. This knowledge gap might lead to suboptimal designs, making it harder to understand, maintain, and evolve a complex model (Hu et al. 2012; Plösch et al. 2008).

Regardless of the lack of guidelines, deciding when and how to use them in a Simulink model is not trivial. Classical indicators naturally arising from principles such as functional decomposition or modularization are hardly applicable. Signals may cross the subsystem hierarchy, and buses are primarily used for graphically encapsulating a set of signals rather than physically reducing the coupling of connected elements. Modelers are faced with additional trade-offs regarding bus usage, such as reducing visual complexity and ease of bulk element manipulation versus the loss of visual information and the risk of bundling logically unrelated signals. In addition, modelers have to organize buses (i.e., ordering and nesting of signals and choosing start and endpoints) to avoid buses that are too "clunky" (Jaskolka et al. 2021).

To better understand the trade-offs of using Simulink buses in practice, we devised an inductive-deductive research approach in a triangulation fashion (Denzin 2017). We first performed an exploratory survey with Simulink practitioners and analyzed bus usage in 433 open-source Simulink models. From the gathered data, we elicited 22 hypotheses concerning the advantages of bus usage, bus size best practices, and situations when to use them and when to avoid them. We then conducted a confirmatory survey with Simulink practitioners, asking about their agreement with the hypotheses, thus testing their validity.

The majority of the hypotheses gained considerable support in the Confirmatory survey, making them candidates for being generally accepted. Thus, our findings provide an empirically grounded stepping stone that paves the way toward guidelines on Simulink bus usage. Next to supporting the design of large-scale Simulink models from a constructive perspective, such guidelines may further serve as requirements for static analysis tools.

This study was conducted in the context of a research project[3] with an industrial partner, namely Model Engineering Solutions GmbH[4], which develops tools that support various Simulink model analyses. Our collaboration supports them in making an evidence-based and informed decision on whether they should extend their portfolio towards buses and how to extend their modeling guidelines and tool suite.

2. Background

Simulink is a graphical block-oriented modeling environment for simulating and analyzing multi-domain dynamical systems. Moreover, code generation facilities transform a model into a classical programming language, usually C. A Simulink model is a data flow graph whose vertices and edges are different kinds of *blocks* and *signal lines*, respectively. Blocks receive inputs from other blocks through signal lines connected by their *ports*. These inputs are processed and forwarded as outputs, yielding a data flow-oriented model (Duran et al. 2009). Two versions of an exemplary model[5] are shown in Figure 1.

Simulink can display different views of a model and offers three means of abstraction and structuring a model: subsystems, MUXes, and buses. Their proper usage reduces the number of visible elements, which is achieved by encapsulating and hiding their contained elements from the outside. As a benefit, unnecessary details from the current view are hidden (Gerlitz et al. 2015), thus, lowering the visual complexity. Further, these can give context information, making the elements' relationships more explicit. Finally, bulk operations are possible by encapsulating several elements, i.e., copying or moving all encapsulated elements together. A subsystem can be compared to a function, method, or procedure of classical programming languages. The MUX can best be compared with an array and a bus with a struct or a dictionary (Rau 2001). In other words, elements of a MUX must be of the same type and are accessed by an index, while elements of a bus can have mixed types and are accessed by their name.

By default, a bus is only *virtual*, i.e., it does not change a model's functionality if it were to be resolved into its elements. Therefore, it does not affect the simulation of a model or its code generation. However, engineers can change this characteristic to *non-virtual*, which forces the generation of C structs in the generated code, preserving the bus elements' cohesion. This way, all bus signals will be created in simulations, leading to decreased performance.

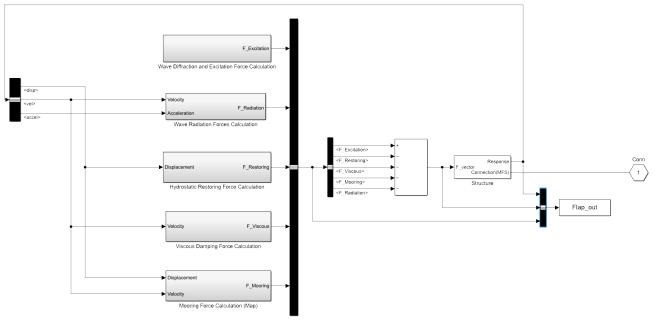
Buses are built by bundling signals in a block called the *bus creator*. This block receives multiple signal lines as inputs and a bus signal as output. To access a bus' elements, the *bus selector* block is used, where the single bus line enters and the user-selected line(s) exit(s). Both are depicted as black bars in Figure 1a. In Figure 1b, we resolved all buses of Figure 1a to show the effect bus usage can have.

On the one hand, without using buses, many new signal lines appear. It is difficult to say which blocks are connected due to the clustering of lines that cross each other at several intersection points. As the number of ports of each block also increases, some blocks need to be resized to carry all of them, highlighting another problem: the sheer number of ports makes it hard, if not impossible, to tell which port of which block is connected to which other port. Additionally, buses allow for signal hierarchies by encapsulating elementary signals and other bus lines.

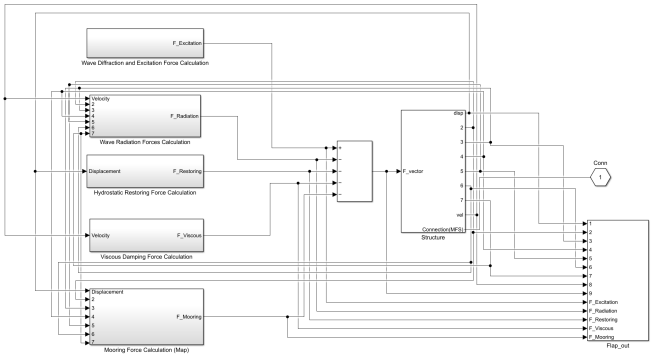
On the other hand, once signals are encapsulated into a bus; the outer view loses information. The number and name of individual signals are no longer visible. Signals of a bus may be unrelated, or some may never be used but still carried along the bus. In addition, modelers have to organize buses through

[3] SimuComp, 01IS18091 BMBF
[4] https://model-engineers.com/en/

[5] We depict the subsystem WECSim_Lib/Body Elements/Rigid Body of the model source/lib/WEC-Sim/WEC_Sim_Lib.slx from the GitHub project https://www.github.com/ratanakso/WEC-Sim-OSU-Temp



(a) One view of a Simulink model, showing blocks connected by signal lines. The black blocks are bus creator blocks (single output on the right) and bus selector blocks (single input on the left). The white and shaded blocks are subsystems, which could be expanded to views of their currently hidden implementation details.



(b) An altered version of the model of Figure 1a, where we artificially resolved all buses and sub-buses up to the elementary signal lines. On the one hand, there are fewer blocks as the bus creators and selectors are removed. On the other hand, various new signal lines, block inputs, and outputs are now part of the diagram. Many of the signal lines cross other lines, and it is hard to discern which blocks are connected by which lines.

Figure 1 Two versions of an exemplary Simulink model showing blocks connected by signal lines. In Figure 1a bus creator and bus selector blocks encapsulate several signal lines. These signal lines can be seen once all buses are resolved in Figure 1b.

bus creators and selectors, increasing the number of required blocks.

3. Study Design

3.1. Research Objective

Our research aims to understand better Simulink bus usage to propose guidelines on how and when to use buses for Simulink practitioners. To this end, our study is driven by the following research questions:

RQ1 What are the advantages of bus usage?

RQ2 When is it appropriate to use buses?

RQ3 When should bus usage be avoided?

RQ4 What are best practices for bus sizes?

3.2. Research Design

For this study, we devised an inductive-deductive research approach in a triangulation fashion (Denzin 2017). The inductive phase is used to develop hypotheses, which serve as guideline candidates. To develop our hypotheses, we conducted an exploratory survey with Simulink practitioners, analyzed open-source Simulink models, and derived them from prior literature. As our developed hypotheses could be invalid (Copi et al. 2006), we test them in a subsequent deductive phase. Here, we conducted a confirmatory survey with domain experts to cross-validate our hypotheses.

3.3. Inductive Phase

In the inductive phase, we followed an exploratory research approach (Shields & Rangarajan 2013) to identify bus usage advantages, disadvantages, and best practices. The phase is composed of three steps; (i) an exploratory survey conducted through a questionnaire composed of open-ended questions,

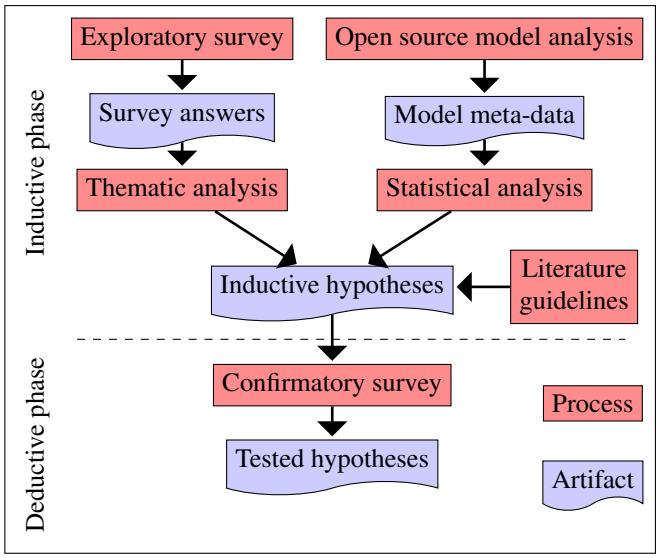


Figure 2 Two major phases of our research workflow: (1) the inductive phase where a survey analysis and model analysis result in the development of hypotheses, and (2) the deductive phase where these hypotheses are tested.

(ii) an analysis of a curated corpus of open-source Simulink models gathered from public repositories, and (iii) a collection of guidelines from the literature (cf. Figure 2). The following sections describe how each of these steps was conducted.

3.3.1. Exploratory Survey We created an online survey with open-ended questions using Google Forms, designed to answer research questions RQ1, RQ2, and RQ3. Three sampling methods were used to reach out to participants. Voluntary response sampling was used through posts in eleven public forum groups from a professional social network[6] related to Simulink

[6] https://www.linkedin.com

Table 1 Overview of the demographic characteristics of the participants of our Exploratory survey.

ID	Application Domain	Role in Organization	Organization size	Years of experience	Skill origin
P1	Automotive	Engineering service	>1000	4	training-on-the-job
P2	Automotive	Technical leader	>1000	9	training-on-the-job
P3	Avionics	Systems engineer	>1000	10	university/education
P4	Automotive	Test Manager	21–100	24	training-on-the-job
P5	Avionics	Chief Software engineer	>1000	18	training-on-the-job
P6	Automotive	Senior Software Engineer	>1000	5	training-on-the-job
P7	Robotics	*No answer*	>1000	7	training-on-the-job
P8	Automotive	Embedded Software Developer	101–1000	4	university/education
P9	*No Domain*	Technical Lead of Functional Safety Software	>1000	6	other
P10	*No Domain*	Software Developer	>1000	10	training-on-the-job
P11	Automotive	Technical Leader	101–1000	6	self-taught
P12	Automotive	Project Engineer	21–100	8	training-on-the-job
P13	Automotive	Senior Software Engineer	>1000	4	training-on-the-job
P14	Automotive	CAE Analyst	>1000	2	self-taught
P15	Automotive	Tool support and design of modeling patterns	>1000	5	university/education
P16	Automotive	SW Developer	101–1000	8	training-on-the-job
P17	Automotive	Product Owner	21–100	3	training-on-the-job
P18	Electronics	MBD Evangelist (self-reported)	1–20	18	university/education

and model-based development (such as *"MATLAB & Simulink"* and *"MATLAB-Simulink and Model-based development"*). The survey was also distributed to the authors' contacts from industry and research (i.e., convenience sampling), which were asked to share it further with relevant peers (i.e., snowball sampling).

We first asked the participants five demographic and two qualification questions. The qualification questions were *Are you aware of the concept of buses in Simulink?* and *Have you ever used a bus in a Simulink model?* They were used to identify whether the participants were fit to continue the survey. The demographic questions and their respective answer options were as follows:

1. What domain of application do you address in your Simulink models? (Energy, Electronics, Avionics, Robotics, Automotive, Health, Other (Boll et al. 2021))
2. What is your role in your organization? (open answer)
3. How many employees does your organization have? (1–20, 21–100, 101–1000, >1000)
4. How many years of experience do you have with Simulink? (open answer)
5. How did you acquire your Simulink skills? (self-taught, university/education, training-on-the-job, other)

Out of 20 responses, one participant never employed buses before the survey, and one mainly gave unclassifiable answers; these two were discarded from the further analysis (data in (Boll et al. 2023)). A summary of the remaining answers to the demographic questions can be seen in Table 1. The reader can use this table to trace the quotes that will be later presented in Section 4 back to the participants' backgrounds through the ID element. The participants' Simulink experience median is 6.5 years, and 8.4 years on average (cf. Figure 3). Moreover, Figure 4 summarizes the distribution of the domains in which the participants are working.

Finally, we asked the participants seven open-ended questions about buses and their application:

ES1: In which situations do you use buses in Simulink models?
ES2: In which situations do you avoid using buses?
ES3: In your opinion, what are the advantages of bus usage in Simulink models?
ES4: Describe a situation in which you experienced inadequate bus usage. How did this impact your work?
ES5: Does a model's size change how you work with buses? If so, please describe.
ES6: Do you differentiate between virtual and non-virtual buses? If yes, in which situations do you use virtual buses, and in which do you use non-virtual buses?
ES7: Under which conditions do you consider refactoring your model from several signals to buses or vice versa?

3.3.2. Thematic Analysis

Having the survey results for the questions ES1 through ES7, we performed a thematic analysis (Braun & Clarke 2021) on the responses. First, we preprocessed the answers by removing those that participants did not develop far enough such that they could contribute to theory building, such as *"No idea"*. Then, we grouped the answers to each question according to their similarities. We created hypotheses for each of these groups according to the respective answers. For instance, we created Hypothesis H8, which states that *buses should be employed to improve efficiency when working on the model*, based on the responses to the question *RQ2 When is it appropriate to use buses?*. The following responses were placed together in the same thematic group, which inspired the creation of Hypothesis 8: *"to gain the ability to perform operations on the whole bus at once in the model"* (P9), *"[it] sometimes is easier to manage a bus structure to communicate software modules instead of updating the interface and adding inputs/outputs to the models"* (P6), *"when it becomes impractical to keep adding inputs outputs"* (P6), and *"consolidating signal inputs to scope"* (P14). Finally, we had a list of hypotheses traceable to the survey answers. In section Section 4, we provide the hypotheses and respective responses that sourced their development.

3.3.3. Open Source Model Analysis

Some information types can be more readily and accurately retrieved from artifacts, especially for quantitative information, *e.g.*, the number of elements bundled in a bus. For answering RQ4, we first collected a set of 4,812 open-source Simulink models on GitHub stemming from 317 different projects (Boll et al. 2023). To this end, we mined GitHub with Google BigQuery (Google 2023) for Simulink projects. While Kalliamvakou et al. warn of the perils of mining GitHub (Kalliamvakou et al. 2014), Boll et al. (Boll et al. 2021) found open-source Simulink models to be diverse and suitable for empirical research, provided one uses an appropriate subset of models, e.g., by removing ad-hoc and toy models. Thus, we constructed a subset of 433 from the original 4,812 models that employed at least one bus (bus creator block or bus output port) for further analysis.

3.3.4. Statistical Analysis

We studied several metrics to quantify how buses are used, which allowed us to differentiate between typical and atypical bus usage. Our rationale for investigating this direction is that although open-source models may not represent *best practice*, typical usage still represents *common practice*, employed by Simulink users. We defined "atypical" bus usages as outliers below the 5th and above the 95th percentile of a metric's distribution and typical usage as everything in between (see Table 2). The metrics we computed per bus are:

- number of elementary signals: we recursively counted all elementary signals of all the sub-busses of a bus (minimum is 1)
- depth of bus: depth of the sub-bus tree (minimum is 1)
- number of sub-buses (minimum is 0)

We then formed hypotheses that atypical bus usage should be avoided or fixed. For instance, Hypothesis H18, which states that a bus should contain less than nine sub-buses, was created after we found that 95% of all busses have less than nine sub-buses. Thus, we consider having nine or more sub-buses as

atypical. In section Section 4, we provide the hypotheses and respective statistical data that sourced their development.

3.3.5. Literature Guidelines From prior literature suggestions (Doerr & Bachmann 2018; Eessaar & Käosaar 2019; Gerlitz et al. 2015), we derived hypotheses on possible refactorings of problematic buses to further answer RQ4. For instance, hypothesis H22, which states that a bus should be made up of at least two elementary signals, was developed from the "superfluous bus signal" smell from Gerlitz et al. (Gerlitz et al. 2015). The value was derived from the 5th percentile of bus usage, which includes two signals. In section Section 4, we provide the hypotheses and respective literature references that sourced their development. These hypotheses fit our personal experience and reasoning, so we included them in the confirmatory survey.

3.4. Deductive Phase

Our goal in this phase was to test our previous findings through triangulation (Denzin 2017). Inductive reasoning allows for the conclusion to be false (Copi et al. 2006); thus, we tested our hypotheses with the help of a confirmatory survey.

3.4.1. Confirmatory Survey After eliciting hypotheses in the inductive phase, we tested them using a confirmatory survey (cf. Figure 2). The idea was to present our hypotheses to Simulink practitioners and ask them to state their level of agreement using a Likert type scale (Likert 1932) of four points (*strongly disagree, disagree, agree, strongly agree*). Additionally, participants could decline to give a rating if they lacked the knowledge or were indecisive on a question (*don't know*). Thus, we employed a forced choice survey (Allen & Seaman 2007), i.e., there was no *neutral* option. We distributed the survey using the same sampling methods and channels as we did with the Exploratory survey, plus among participants of the workshop *Buses in Simulink – A Blessing or a Curse?* organized by the Modeling Guidelines Interest Group (MES 2023). It included the same qualification and demographic questions as in the Exploratory survey, described in Section 3.3.1. The survey received 36 responses, of which we excluded six: one from a participant answering twice, another answering every question only with *agree*, and four participants stated having no experience with Simulink buses. Finally, 30 responses remained for further analysis. Participants have a median of 8 years of experience and an average of 10.8 years (cf. Figure 3). The participants' domains are summarized in Figure 4. Besides, the participants of the confirmatory survey show a similar diversity and distribution as the participants of the Exploratory survey (more details can be found in our data set (Boll et al. 2023)).

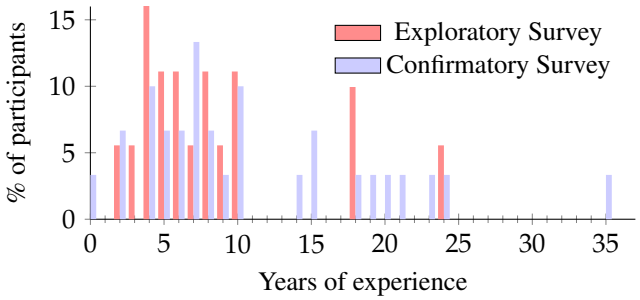


Figure 3 Distribution of the years of experience in working with Simulink of the participants of our surveys. The respondent with 35 years of experience clarified their answer "I worked on System Build, before Simulink existed."

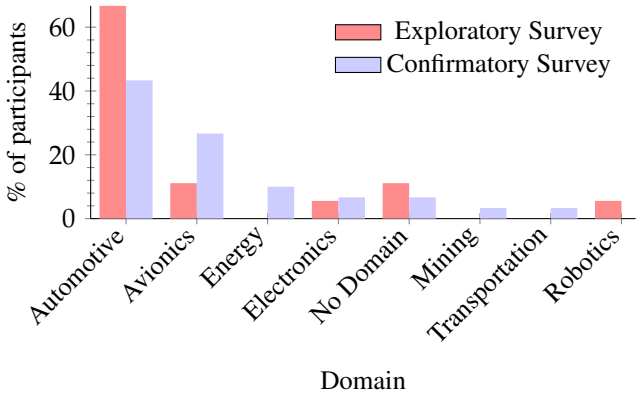


Figure 4 Distribution of the domains in which the participants of our surveys are working.

4. Results

4.1. Inductive phase results

Here, we present the 22 hypotheses developed during the Inductive phase of our study. We derived the hypotheses from multiple sources of evidence, namely the responses of the 18 participants of the Exploratory survey, the Open source model analysis, and the Literature guidelines. In the following, we de-

Table 2 Summary of the distributions of our bus size metrics. We define "atypical" bus usage as below the 5th percentile or above the 95th percentile (numbers marked in pink). The minimum bus depth and the minimum number of sub-buses equal the fifth percentiles (the minimal possible values), so there is no lower cut-off. All metrics are long-tail distributions; thus, the averages are higher than the medians and include extreme outliers.

Per bus	min	p_5	p_{25}	med	p_{75}	p_{95}	max	avg
# of elementary signals	1	2	3	5	12	37	1392	15.49
bus depth	1	1	1	1	2	3	7	1.44
# of sub-buses	0	0	0	0	1	8	291	2.15

scribe these hypotheses and provide their source (e.g., excerpts from the participants' answers).

4.1.1. RQ1 What are the advantages of bus usage?

This section presents the hypotheses that address RQ1. They were elicited from the Exploratory survey; in particular, they were derived from the answers to question ES3.

Hypothesis H1. Buses reduce visual clutter in models. Generally, the participants value the very purpose of a bus to bundle together signals into a composite signal (*"combine signals" (P1)*, *"less signal lines" (P8)*) to diminish the number of visible elements (i.e., lines and interfaces). In particular, models with fewer lines are considered to be cleaner, as indicated by notions such as *"cleaner models" (P10)*, *"cleaner interface" (P3)*, or *"more clear models" (P16)*, and to provide a better overview (*"better overview with less signal lines" (P17)*, *"much better overview in the models, less signal lines" (P15)*, *"better overview" (P9)*). In addition, some participants even mentioned that the reduction of the number of lines through using buses fosters *"simplification and flexibility" (P6)*, and that buses may help in *"reducing complicatedness in complex models" (P18)*.

Hypothesis H2. Buses are useful to provide engineers with information regarding signal relatedness through the model design. Buses can be used to convey information to engineers, who make assumptions about the data flow based on the signal visual flow (Gerlitz et al. 2015). Bundling signals in a bus suggests that these signals share a relation. Participants note that buses *"show constructively, which signals belong together (coming from the same source and mostly be used together)" (P4)* and that they enable a *"logical grouping of signals" (P9)*.

Hypothesis H3. Buses are useful to facilitate signal manipulation. A bus is a composite signal type that allows element access through names (*i.e.,* like a struct or hash table), as opposed to index-based access. Thus, the elements' order is irrelevant. Such characteristic eases signal selection, e.g., *"signal routing is made easier, and I don't have to care about the signals order like in MUX" (P12)*. When the signals are grouped in a bus, they can be shared between subsystems with less effort, as fewer diagrammatic operations are needed: *"less manual effort to connect signals one by one" (P13)*. Engineers can also *"perform operations on the whole bus at once" (P9)*, and can *"handle a set of signals" (P11)*.

Hypothesis H4. Buses are useful for configuring simulation and code generation. Non-virtual buses can be used for configuring model simulation and *"generating structs in code" (P18)* (i.e., C-structs), which offers *"good cohesion with traditionally generated code through shared header files" (P5)*, as the *"structure in the code if non-virtual buses are used" (P17)* is preserved.

4.1.2. RQ2 When is it appropriate to use buses?

This section presents hypotheses derived from the Exploratory survey addressing RQ2. More specifically, they stem from the answers to ES1, ES5, ES6, and ES7 questions.

Hypothesis H5. Buses should be employed when structuring diagrams. Many participants mentioned the architectural quality of buses. They noted that buses can be used *"as structure" (P1)*, *"to create structures" (P11)*, and to *"model architecture" (P9)*. It is preferable to have a *"structure of signals [for] too many scalar signals" (P17)*, or for *"multiple signals with different data types as structure" (P1)*. They can also facilitate encapsulation of subsystem communication and be viewed *"mainly as interfaces" (P7)*, or as an *"interface definition" (P17)* itself.

Hypothesis H6. Buses should be employed when grouping related signals (*i.e.,* used by the same subsystem or function). The relatedness of candidate signals is a trigger for bus usage. Participants advise to *"couple related signals" (P3)*, *"to structure data that belong together" (P10)*, and employ buses *"when grouping similar signals" (P14)*, or *"when they have a strong dependency on each other" (P10)*. The modeled physical entity also influences bus usage. Some participants use buses when they *"reflect [a] real data network" (P3)*, and even more explicitly choose to *"combine CAN data and status signals" (P13)*. Two signals can be considered related if the same part of the system uses them, or they share the same function: *"if more than 2 data signals need to go through the same operation" (P2)*. In this way, buses *"transport signals, which belong together topically, between subsystems" (P8)*, or are used for *"exchanging large signals between sub-assemblies" (P14)*. More generally, it may be enough if signals take a similar flow through the model. Participants use buses *"to gather large groups of signals and lead them through the model" (P15)*, *"to bunch the signals moving between modules" (P13)*, and *"when [signals] are used in a combination at multiple locations" (P10)*. One participant explicitly mentioned that a signal's value might not be critical and recommends *"routing grouped constants and/or variables" (P18)*.

Hypothesis H7. Buses should be employed to improve model comprehensibility. Upon a high density of signal lines, the understandability of the diagram becomes impaired. A participant noted that the *"comprehensibility of the model" (P4)*

could be improved, as *"too many signal lines could be overwhelming" (P4)*. In this way, buses can be used *"to simplify drawings by decreasing the number of crossing signals" (P10)* and are *"consolidating signals, to get a better signal flow overview" (P9)*. In other words, buses should be used as the *"model gets confusing due to too many signal lines" (P8)*. Another important aspect of buses is *"to simplify interfaces" (P6)*, by *"reducing interface size" (P18)*. This aspect can be viewed as fewer signals flowing into inports or from outports, and thus buses *"are also used to reduce the number of ports between the subsystems" (P16)*. Buses generally also simplify the layout of subsystems significantly. Both in terms of understanding and creating the layout. Getting a good routing in Simulink with many lines is effort intensive.

Hypothesis H8. Buses should be employed to improve efficiency when working on the model. Signals can be manipulated in bulk when grouped in a bus. Engineers thus want *"to gain the ability to perform operations on the whole bus at once in the model" (P9)*. Buses also save the time required to update interfaces between communicating software modules. Participants described this as *"[it] sometimes is easier to manage a bus structure to communicate software modules instead of updating the interface and adding inputs/outputs to the models" (P6)*, or *"when it becomes impractical to keep adding inputs/outputs" (P6)*. Finally, *"consolidating signal inputs to scope" (P14)* is also facilitated by buses.

Hypothesis H9. Buses should be employed when defining C-structs to be created in the auto-generated code. Buses marked as non-virtual trigger the code generator to transform the containing signals into C-structs. Several participants mentioned this feature, as they note buses are *"also useful to interface with C-structs" (P6)*, and *"code generation of structs" (P18)*, or even the very general remark: *"all models use buses to generate/access to expected C-structs in the code" (P16)*.

4.1.3. RQ3 When should bus usage be avoided?

The hypotheses presented in this section address RQ3 and were derived from the Exploratory survey. More precisely, they were developed from the answers to questions ES2, ES4, ES5, ES6, and ES7.

Hypothesis H10. Buses should not be used when the number of candidate signals for a bus is low. Engineers may avoid using buses if the number of candidate signals one considers bundling in a bus is too low. This is usually the case for *"models with few signals" (P18)*, or *"models with simple interfaces" (P6)*. The participants considered different numbers of signals as too low, though: in *"small models [with] not so many signals (many being less than 8 or a bit more)" (P4)*, when the *"number of signals is ≤ 3" (P8)*, or even *"when it's only one signal" (P14)*. Too few signals in a bus can even lead to bus refactoring by bus dissolution from *"bus to signals: when only single signals are used in deeper model levels" (P10)*. Hypothesis H22 revisits this aspect and suggests a concrete number for "too low".

Hypothesis H11. Buses should not be used when signals have the same data type; a MUX should be employed instead. This hypothesis is similar to one of the very few guidelines concerning buses of MAB and MISRA (Misra 2023; MathWorks Advisory Board 2020). One should use MUXes for the same typed data and buses for mixed typed data, which one participant also stated as in the case of *"same data type signal" (P1)*.

Hypothesis H12. Buses should not be used when the candidate signals are unrelated. This hypothesis is the reverse of Hypothesis H6, and most participants thus answered with a reversed response. One participant noted that *"if the combination of signals into a bus is unpractical, e.g., signals come from/go to different subsystems, signals don't belong together" (P16)*.

Hypothesis H13. Non-virtual buses should not be used when they affect code generation negatively. To optimize the code generation, non-virtual buses should be avoided, i.e., there is an *"autocode optimization need" (P12)*. Unlike non-virtual buses, virtual buses reduce memory requirements by accessing and storing data non-contiguously, *"to avoid memory conception" (P11)*.

Hypothesis H14. Buses should not be used when testing models; signals should be used instead. Doerr et al. mentions that, in testing, all signals of a bus will be created, even when they are not used elsewhere (Doerr & Bachmann 2018). This superfluous signal creation adds to the cognitive load of the tester. Two participants also noted *"a to be unit tested (referenced) subsystem shall only get the inputs it needs, to minimize testing and analysis effort" (P9)*, and *"for testing models, signals are better" (P11)*.

Hypothesis H15. Buses should not be used when conveying signal usage information. When located outside buses, signals and their respective names become more visible and can convey information to the engineer. This way, one can be explicit and use a lower abstraction level, e.g., elementary signals. It is also easier to discern which signals of a bus are used (Doerr & Bachmann 2018), or more concretely *"to show on which signals the sub-model really relies on" (P10)*. Participants do not use buses *"if [they] want to model very clearly how submodules interact with each other (through which signals)" (P6)*, and *"don't often remove buses, but if it benefits understanding at the higher level, [they] can select signals from the bus and route as scalars into the subsystems" (P18)*.

4.1.4. RQ4 What are best practices for bus sizes?

The hypotheses in this section were developed either from analyzing publicly available Simulink models or from suggested refactorings of (Eessaar & Käosaar 2019) and (Doerr & Bachmann 2018).

Hypothesis H16. If a bus is becoming too big, it should be split up into its sub-buses. This hypothesis states a refactoring proposal for big buses. If a big bus consists of sub-buses, it could be resolved into its sub-buses (a more radical step would be exposing even the sub-buses recursively as elementary signal lines). This refactoring could aid the understanding and handling, as smaller, more manageable buses can now be worked on, and more information is explicit, see Hypothesis H15.

Hypothesis H17. A bus should contain less than 38 elementary signals. In our analysis of public Simulink models, we found that 95% of all busses have less than 38 elementary signals (see Table 2). Thus, we consider having 38 or more

signals as atypical. A possible refactoring is splitting the bus into sub-buses; see Hypothesis H16.

Hypothesis H18. A bus should contain less than nine sub-buses. We found that 95% of all busses have less than nine sub-buses (see Table 2). Thus, we consider having nine or more sub-buses as atypical. A possible refactoring is extracting some sub-buses (see Hypothesis H16).

Hypothesis H19. A bus should be nested less than five bus layers deep. We found that 95% of all busses are nested less than four layers deep (see Table 2). Thus, we consider having four or more layers of hierarchy as atypical. There was a typo in our questionnaire: our hypothesis should have stated "less than four bus layers deep"; it is thus less strong and may have led to a higher agreement rate. Possible refactorings are splitting the bus into sub-buses, see Hypothesis H16, or flattening the sub-bus hierarchy by resolving sub-buses into their elementary signals.

Hypothesis H20. When less than 50% of a bus' signals are used, it should be split up. One intuitive refactoring proposal for splitting up a bus is excluding unused signals. We suggest refactoring buses in which too many signals are unused, as they inflate the bus, making it harder to understand and handle. One participant in the Exploratory survey stated the following suggestion: *"when only single elements of the bus are necessary in the sub-models" (P10).*

Hypothesis H21. If a bus is too small, its elementary signals should be used instead. Buses that group up a tiny number of elements could add unnecessary abstraction (see Hypothesis H15).

Hypothesis H22. A bus should be made up of at least two elementary signals. This hypothesis brings a value for the "superfluous bus signal" smell from Gerlitz et al. (Gerlitz et al. 2015). The value was derived from the 5th percentile of bus usage, which includes two signals (cf. Table 2). Following this hypothesis, only buses using single elements must be refactored. As the minimum number of bus elements is one, this hypothesis is not particularly strong, but it gives a concrete number for the imprecisely defined "low" of Hypothesis H10.

4.2. Deductive phase results

The results of the Confirmatory survey are depicted in Figure 5. The numbers are given as percentages for the agreement level of the 30 respondents of the Confirmatory survey. We excluded the "don't know" answers participants gave, which we show separately on the right in gray color, as percentages of all responses. We did this because "don't know" responses cannot be rated on our forced-choice Likert type scale, see Section 3.4.1.

5. Discussion

In our discussion of study results, we first focus on the Confirmatory survey results, then discuss the results' applicability for industry and academia and finally consider potential threats to validity.

5.1. Confirmatory survey results discussion

To discuss the Confirmatory survey results, we first introduce two dimensions, *consensus* and *mean agreement*, as criteria to analyze a hypothesis' acceptance. Consensus (Tastle & Wierman 2007) measures the dispersion within an ordinal scale's answers. It is defined between 0 (complete dispersion between respondents) and 1 (complete agreement between respondents). The mean agreement is computed by converting the ordinal scale into an equidistant interval scale.[7] These values for each hypothesis are depicted in Figure 6, similarly to (Vogelsang et al. 2020). The x-axis shows the mean agreement value towards the hypotheses, the y-axis shows the consensus level of the participants. Color and shape disclose the median agreement for each hypothesis. In this chart, *don't know* answers were removed. For instance, hypotheses with a lower consensus are more controversial. They are located in the lower part of the chart (e.g., H4, H8, H9). These may need some clarification to be universally accepted or apply only to specific contexts. On the other hand, hypotheses located on the upper part of the chart (e.g., H1, H2, H7, H20) had higher consensus, probably because they are more generalizable and were well understood by the participants. For the mean agreement, hypotheses located next to the horizontal extremes have more *strong agreement* or *strong disagreement*. For instance, hypothesis H1 is located very much to the right side, with 73% of *strongly agree*. In the following, we analyze the hypotheses, comparing them according to the responses received.

5.1.1. Hypotheses with high agreement rate

All hypotheses from RQ1, RQ2, and RQ4 had a good agreement rate (i.e., 73% on average). The hypotheses of RQ1 and RQ2 concern the advantages of buses and usage situations; thus, practitioners recognize buses' positive attributes and use cases. Perhaps the cut-off numbers in RQ4's hypotheses could have been stricter, as the agreement rate to bus size hypotheses was fairly high. In any case, bus sizes in the upper and lower fifth percentiles of open source model usage (cf. Table 2) were considered problematic by most respondents.

5.1.2. Hypotheses with high disagreement rate

Four hypotheses received more disagreement than agreement: H10, H11, H14, and H15. They all address RQ3, which focuses on *situations when buses should be avoided*. Respondents could not agree on the hypotheses with a similar consensus to RQ2, which focuses on *when it is appropriate to use buses*. For instance, H10 was created based on the statements of six participants with a median of 7.5 years and an average of 10.5 years of experience. However, 55% of the participants of the Confirmatory survey disagreed with it. One possible reason for this outcome is the way the hypothesis was formulated (i.e., *"the number of candidate signals for a bus is low."*) since most participants agreed that buses containing only two signals are too few (H22). Interestingly, H11 was the second most disagreed hypothesis, despite being elicited from the official guidelines. This fact may show the general need for founding or revising guidelines empirically based on actual practitioners' needs. Hypothesis H14 was derived from two survey participants, and we also

[7] We are aware, that the responses are scaled ordinally and calculating a mean value for them is refrained from. Computing them for hypotheses with the same median values can still give an informative order of the level of agreement, though. We do not interpret mean values in our discussion.

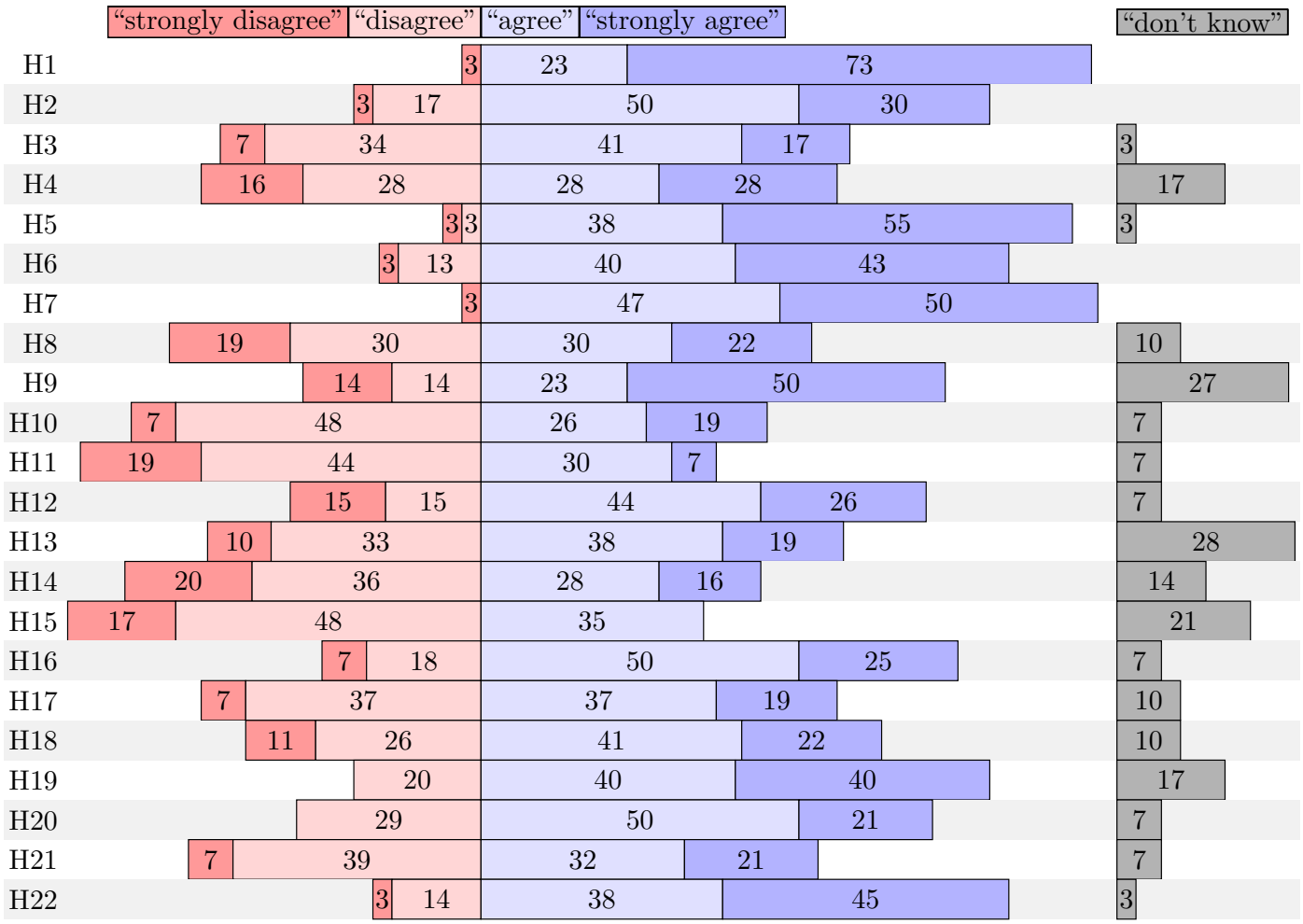


Figure 5 Confirmatory survey results. The colored bars in the middle show each hypothesis' agreement rate from the Confirmatory survey. The hypothesis IDs are denoted on the left side. The dark pink bars show the percentages of strong disagreement, light pink bars give the percentages of disagreement, light blue for agreement, and dark blue for strong agreement. Here, the "don't know" answers are excluded. They are shown on the right side with gray bars. Their numbers describe percentages of all responses with and without an agreement rating.

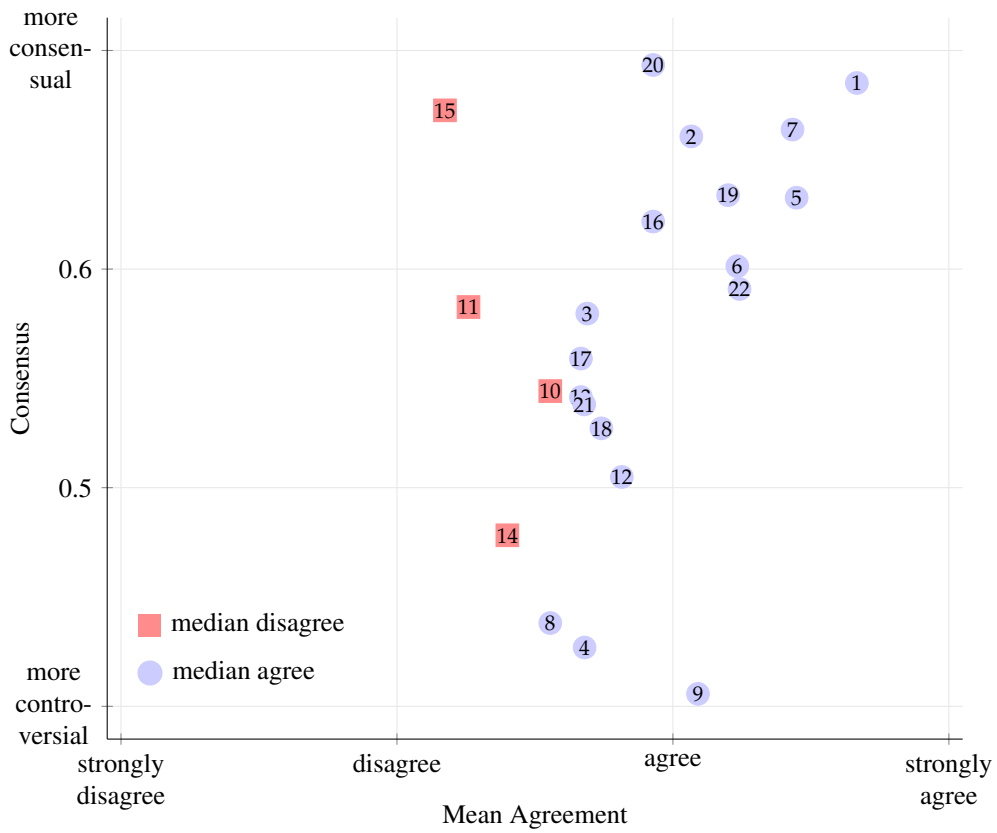


Figure 6 Mean agreement and consensus of hypotheses. Hypothesis H13 is located behind hypothesis H21.

find reference to it in the literature (Doerr & Bachmann 2018). Perhaps, a rewritten hypothesis with more information, such as *thus, diminishing testing efforts* would have had a higher agreement rate. In H15, the least agreed hypothesis with only 35% and the only hypothesis without a single *strongly agree*, the wording of *"conveying signal usage information"* similarly could be open to misinterpretation. Disagreement with H15 was rather consensual, suggesting the opposite of H15 may find more acceptance: *buses should be used to convey signal usage information.* These hypotheses are good candidates for future work, where we try to understand the reasons for low adherence.

5.1.3. Hypotheses with high "don't know" rate Overall, the ratio of participants answering with *don't know* was low (9.2%), from which we conclude that our hypotheses are understandable and practitioners indeed have the knowledge and an opinion about them. Two hypotheses (H9 and H13) had more than 25% of *don't know* responses, and both are concerned with non-virtual buses, which we assume is a Simulink feature many engineers do not employ or have little knowledge of. This phenomenon could also explain why H9 achieved the lowest consensus in our study (cf. Figure 6). Despite many respondents being unable to provide a stance, the ones who did mostly agree with these hypotheses. Similarly, H4, where 56% of the respondents have agreed, shows that buses are used relatively little for code generation. Having notions of "simulation" and "code generation" in the same sentence could have confused the respondents due to their relation, influencing the high rate (17%) of *don't know* answers.

5.2. Applicability of our findings

This section discusses our findings' applicability to industry and academia.

5.2.1. Impact for Industry The bus element, when properly employed, lowers an engineer's cognitive demand by increasing internal model quality (ISO/IEC 25010 2011). This may affect static properties, such as analyzability, modifiability, and testability. Poor internal quality, and smells (Fowler & Becker 1999) can contribute to increased technical debt (Tufano et al. 2015). In this matter, hypotheses H6 and H9 can serve as new guidelines on when to use buses. H12 and H13 guide when to avoid bus usage. H16 and H21 give concrete refactoring guidelines for buses with suboptimal signal amounts. Additionally, H17–H20 and H22 present concrete and empirically founded antipatterns. Detection of these cases is a good candidate to be implemented in static analysis tools and resolved through one of the refactorings proposed in H16 or H21.

For Model Engineering Solutions GmbH (MES), our industrial partner, the findings became the input for creating a static analysis tool prototype to give engineers hints on where buses can be used sensibly. This prototype represents the first step before incorporating these functionalities into their commercial tool. Overall, their perception is that buses are not used as much as they should; thus, the best practices for recommending when to use buses are the most interesting ones from their point of

view. Our findings are also used in their Simulink teaching classes for system engineers.

5.2.2. Impact for Academia Hypotheses H1–H5, H7, and H8 empirically validate knowledge about the basic properties of bus usage. These describe conceptual grounding knowledge relevant to engineers new to Simulink. They describe benefits and define use contexts. This knowledge can also be incorporated into higher education course programs. Our findings contribute to the body of knowledge of design patterns in visual programming languages (Yazar 2014) with a strong focus on their graphical properties. Dataflow programming languages (Sousa 2012) (e.g., ASCET-DEVELOPER (ETAS 2023)) bearing similar types of elements to the Simulink buses yields benefits from the results of our research. Some hypotheses and proposed guidelines may seem trivial, but they still help beginners by making tacit knowledge explicit and advanced practicioners by making the knowledge referable and consultable. We propose basing guidelines on empirical research, as H11 (based on a published guideline) was one of the hypotheses participants disagreed with the most.

5.3. Threats to Validity

During the inductive phase of our research, we used an online survey with open-ended questions, which could lead to misinterpretations of questions. To mitigate this threat, we carefully reviewed the survey before its execution, reviewed its responses after its execution, and discarded the ones that clearly showed some misunderstanding or did not provide helpful information for theory building. We posted our surveys in professional public forums (*i.e.,* voluntary response sampling) and asked our peers to share them with relevant colleagues (*i.e.,* snowballing sampling). Therefore, we had little control over who responded to our survey. This threat was mitigated through five demographic questions and two qualification questions, where the respondents were asked whether they had knowledge and experience using buses. We excluded respondents who never used buses in Simulink models from both surveys. With this exclusion criterion, we possibly also have excluded individuals who got a detailed explanation on why they should never employ buses. We presume, that most participants who have never used a bus, also don't know much about them, though. In any case, only one participant in the exploratory survey and three participants in the confirmatory survey may have been affected as they were excluded because of this criterion.

Webb (Webb et al. 1999) notes that a strength of triangulation is the cross-validation of hypotheses by different methods. Specifically, "When a hypothesis can survive the confrontation of a series of complementary testing methods, it contains a degree of validity unattainable by one test within the more constricted framework of a single method". However, once the different methods do not give converging results for a hypothesis, their degree of validity becomes questionable. This is why we only consider a hypothesis cross-validated by our confirmatory survey if its median level is in agreement, i.e., blue circles in Figure 6. Still, this does not guarantee their validity – and symmetrically neither, that hypotheses H10, H11, H14, and H15 do not hold.

6. Conclusion

We proposed 22 hypotheses concerning Simulink bus elements, covering the advantages of bus usage, how and when to apply buses, and when they should be avoided. Eighteen of these hypotheses were agreed upon, while only four hypotheses received more disagreement. We used three methods to elicit our hypotheses: an open questions survey applied to experienced practitioners, an investigation of structuring decisions by studying real-life Simulink models, and a collection of guidelines from prior literature. In the second step, we applied a confirmatory survey to assess practitioners' agreement with our initial findings. Our findings help close the knowledge gap on these elements, which has not yet received enough attention from the scientific community. They serve as starting point for guidelines for Simulink practitioners and the development of smell detectors. In future work, we want to empirically compare the usage of buses to other abstraction and structuring methods of Simulink, namely the subsystem and MUX.

Acknowledgments

This work has been supported by the German Ministry of Research and Education (BMBF) within project SimuComp (Simulink Architecture Comprehension and Analysis) under grant 01IS18091.

References

Allen, I. E., & Seaman, C. A. (2007). Likert scales and data analyses. *Quality progress*, *40*(7), 64–65.

Boll, A., Amorim, T., Bachmann, F., Kehrer, T., Vogelsang, A., & Pohlheim, H. (2023). *Data set and calculations of this study.* doi: 10.5281/zenodo.8011478

Boll, A., Brokhausen, F., Amorim, T., Kehrer, T., & Vogelsang, A. (2021). Characteristics, potentials, and limitations of open-source Simulink projects for empirical research. *Software and Systems Modeling*. doi: 10.1007/s10270-021-00883-0

Braun, V., & Clarke, V. (2021). *Thematic analysis: A practical guide*. SAGE Publications Ltd.

Copi, I., Cohen, C., & Flage, D. (2006). *Essentials of logic*. Routledge.

Dajsuren, Y. (2015). *On the design of an architecture framework and quality evaluation for automotive software systems* (Unpublished doctoral dissertation). Department of Mathematics and Computer Science, Technische Universiteit Eindhoven.

Dajsuren, Y., van den Brand, M. G., Serebrenik, A., & Roubtsov, S. (2013). Simulink models are also software: Modularity assessment. In *9th International ACM Sigsoft Conference on Quality of Software Architectures (QoSA)* (pp. 99–106).

Denzin, N. K. (2017). *Sociological methods*. Routledge. doi: 10.4324/9781315129945

Doerr, H., & Bachmann, F. (2018). *Analysis and improvement of model architectures for safety related systems* (Tech. Rep.). SAE Technical Paper.

Duran, F., Atacak, İ., & Ömer Faruk Bay. (2009). Simulink stateflow for algorithm learning. *Procedia - Social and Behavioral Sciences*, *1*(1), 554–558. doi: 10.1016/j.sbspro.2009.01.100

Eessaar, E., & Käosaar, E. (2019). On finding model smells based on code smells. In R. Silhavy (Ed.), *Software engineering and algorithms in intelligent systems* (pp. 269–281). Cham: Springer International Publishing.

ETAS. (2023). *ASCET-DEVELOPER – Model-based design and auto c-code generation for embedded systems.* Retrieved from https://www.etas.com/en/products/ascet-developer.php (Accessed: 06.06.2023)

Fowler, M., & Becker, P. (1999). *Refactoring: Improving the design of existing code*. Addison-Wesley.

Gerlitz, T., Tran, Q. M., & Dziobek, C. (2015). Detection and Handling of Model Smells for MATLAB/Simulink models. In *MASE@ MoDELS* (pp. 13–22).

Google. (2023). *BigQuery.* Retrieved from https://cloud.google.com/bigquery (Accessed: 06.06.2023)

Hu, W., Loeffler, T., & Wegener, J. (2012). Quality model based on ISO/IEC 9126 for internal quality of MATLAB/Simulink/Stateflow models. In *IEEE International Conference on Industrial Technology* (pp. 325–330).

ISO/IEC 25010. (2011). *ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.*

Jaskolka, M., Pantelic, V., Wassyng, A., & Lawford, M. (2020). Supporting modularity in Simulink models. *arXiv preprint arXiv:2007.10120*.

Jaskolka, M., Pantelic, V., Wassyng, A., Lawford, M., & Paige, R. (2021). Repository Mining for Changes in Simulink Models. In *ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS'21)* (p. 46-57). doi: 10.1109/MODELS50736.2021.00014

Jaskolka, M., Scott, S., Pantelic, V., Wassyng, A., & Lawford, M. (2020). Applying modular decomposition in Simulink. In *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW'20)* (pp. 31–36).

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., & Damian, D. (2014). The promises and perils of mining github. In *Proceedings of the 11th working conference on mining software repositories* (pp. 92–101).

Liggesmeyer, P., & Trapp, M. (2009). Trends in embedded software engineering. *IEEE software*, *26*(3), 19–25.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, *22*(140), 1–55.

MathWorks Advisory Board. (2020). *Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow – Version 5* (Tech. Rep.). The MathWorks, Inc. Retrieved from https://de.mathworks.com/content/dam/mathworks/mathworks-dot-com/solutions/mab/mab-control-algorithm-modeling-guidelines-using-matlab-simulink-and-stateflow-v5.pdf (Accessed: 06.06.2023)

MES. (2023). *Modeling Guidelines Interest Group (MGI-Group).* Retrieved from https://model-engineers.com/en/academy/mgigroup (Accessed: 06.06.2023)

Misra. (2023). *MISRA AC SLSF* (Tech. Rep.). MISRA. Retrieved from https://www.misra.org.uk/product/misra-ac-slsf/ (Accessed: 06.06.2023)

Pantelic, V., Postma, S., Lawford, M., Jaskolka, M., Mackenzie, B., Korobkine, A., ... Wassyng, A. (2017). Software engineering practices and Simulink: bridging the gap. *International Journal on Software Tools for Technology Transfer*, *20*, 95-117.

Plösch, R., Gruber, H., Hentschel, A., Körner, C., Pomberger, G., Schiffer, S., ... Storck, S. (2008). The EMISQ method and its tool support-expert-based evaluation of internal software quality. *Innovations in Systems and Software Engineering*, *4*(1), 3–15.

Popoola, S., & Gray, J. (2021). Artifact Analysis of Smell Evolution and Maintenance Tasks in Simulink Models. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (pp. 817–826).

Rau, A. (2001). On model-based development: decomposition and data abstraction in simulink. *Gesellschaft fuer Informatik, FG*, *2*(1).

Shields, P. M., & Rangarajan, N. (2013). *A playbook for research methods: Integrating conceptual frameworks and project management*. New Forums Press.

Shrestha, S. L., Chowdhury, S. A., & Csallner, C. (2022). SLNET: A Redistributable Corpus of 3rd-party Simulink Models. *arXiv preprint arXiv:2203.17112*.

Sousa, T. B. (2012). Dataflow programming concept, languages and applications. In *Doctoral Symposium on Informatics Engineering* (Vol. 130).

Tastle, W. J., & Wierman, M. J. (2007). Consensus and dissention: A measure of ordinal dispersion. *International Journal of Approximate Reasoning*, *45*(3), 531–545.

Tufano, M., Palomba, F., Bavota, G., Oliveto, R., Di Penta, M., De Lucia, A., & Poshyvanyk, D. (2015). When and Why Your Code Starts to Smell Bad. In *IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)* (Vol. 1, p. 403-414). doi: 10.1109/ICSE.2015.59

Vanherpen, K., Denil, J., Vangheluwe, H., & De Meulenaere, P. (2015). Model transformations for round-trip engineering in control deployment co-design. *SpringSim (TMS-DEVS)*, *920*, 55–62.

Vogelsang, A., Eckhardt, J., Mendez, D., & Berger, M. (2020). Views on quality requirements in academia and practice: commonalities, differences, and context-dependent grey areas. *Information and Software Technology*, *121*, 106253. doi: 10.1016/j.infsof.2019.106253

Webb, E. J., Campbell, D. T., Schwartz, R. D., & Sechrest, L. (1999). *Unobtrusive measures* (Vol. 2). Sage Publications.

Whalen, M. W., Murugesan, A., Rayadurgam, S., & Heimdahl, M. P. (2014). Structuring Simulink models for verification and reuse. In *Proceedings of the 6th international workshop on modeling in software engineering* (pp. 19–24).

Yazar, T. (2014, December). Design of dataflow. *Nexus Network Journal*, *17*(1), 311–325. doi: 10.1007/s00004-014-0222-8

About the authors

Tiago Amorim is a guest post-doctoral research fellow at the University of Cologne (DE), where he also obtained his Ph.D. Before his doctorate, he worked as a research assistant at the Technical University of Berlin (DE) and the Fraunhofer Institute for Experimental Software Engineering (DE). His research interests are model-based systems engineering, process modeling, software engineering, and empirical research. He is currently looking for a position as a researcher in academia or industry. You can contact the author at amorim@cs.uni-koeln.de or visit https://tbuarque.github.io/.

Alexander Boll is a doctoral student at the University of Bern and has been part of the Software Engineering Group since 2022. Before that, he studied computer science at Humboldt-Universität zu Berlin, where he started his doctoral studies. His research interest is Open Science in the modeling community. You can contact the author at alexander.boll@inf.unibe.ch or visit https://model-engineers.com.

Ferry Bachmann works at Model Engineering Solutions GmbH (MES). He develops professional tools to improve and simplify model-based software development with Simulink. The focus is on automatic layout, refactoring support, and complexity analysis and reduction of Simulink models. You can contact the author at ferry.bachmann@model-engineers.com or visit https://model-engineers.com.

Timo Kehrer is a professor at the Institute of Computer Science of the University of Bern (CH), chairing the Software Engineering Research and Teaching Group. With a PhD from the University of Siegen (DE) and after holding a post-doctoral research fellow position at Politecnico di Milano (IT), he previously was an assistant professor at the Department of Computer Science at Humboldt-Universität zu Berlin (DE). Kehrer has active research interests in various fields of software engineering, including model-driven methods which enable formal reasoning and simulation, and which facilitate the automated transition between informally sketched requirements and implementations. You can contact the author at timo.kehrer@inf.unibe.ch or visit https://seg.inf.unibe.ch/.

Andreas Vogelsang is a Full Professor of Software and Systems Engineering at the Institute of Computer Science at the University of Cologne. His research focuses on requirements engineering, model-based systems engineering, and software engineering with and for machine learning. He is a Junior Fellow of the German Informatics Society (GI), and he was awarded "Young Scientist of the Year" by academics and the German Association of University Professors and Lecturers (DHV). You can contact the author at vogelsang@cs.uni-koeln.de or visit https://cs.uni-koeln.de/sse.

Hartmut Pohlheim been driving forward the quality assurance of software models for the automotive industry for more than 20 years. He holds a doctorate in technical cybernetics and automation engineering from the Technical University of Ilmenau and is considered one of the most distinguished experts in model-based software development. After graduating, he worked in the research department of Daimler AG, where he focused on the optimization of technical applications and the visualization of complex systems. Since 2008 Hartmut Pohlheim has been Managing Director of Model Engineering Solutions (MES) and is responsible for technology development as Chief Technology Officer (CTO). MES provides solutions in various development chains and highly-automated cloud environments for safe controller software. The main focus is static model analysis and model improvement, primarily in MATLAB Simulink, the automotive industry's leading development platform. You can contact the author at pohlheim@model-engineers.com or visit https://model-engineers.com.