CHALMERS
UNIVERSITY OF TECHNOLOGY

What do Users Expect of Bidirectional Transformations?

Citation for the original published paper (version of record):
Wohlrab, R., Anjorin, A., Shankar Mishra, A. (2020). What do Users Expect of Bidirectional Transformations?. Journal of Object Technology, 19(2): 4:1-4:19.
http://dx.doi.org/10.5381/jot.2020.19.2.a4

Journal of Object Technology
Published by AITO — Association Internationale pour les Technologies Objets
http://www.jot.fm/

What do Users Expect of Bidirectional Transformations?

Rebekka Wohlrab[a] Anthony Anjorin[b] Arjya Shankar Mishra[c]

a. Systemite AB and Chalmers University of Technology, Gothenburg, Sweden

b. Paderborn University, Paderborn, Germany

c. Fastec GmbH, Paderborn, Germany

Abstract A bidirectional transformation (bx) is a means of maintaining the consistency of two or more related sources of information. Existing bx frameworks attempt to capture what it means for a bx to be *well-behaved* by proposing various laws. In many cases, however, it is still unclear if such laws really capture "desirable" behaviour from a user's perspective. We argue that knowing most users' expectations is not only useful as input for designing, extending, and improving existing bx frameworks and tools, but also when attempting to choose the most suitable bx tool for a specific application scenario and group of users. In addition, being aware of intuitive expectations that cannot be fulfilled by a chosen bx tool (perhaps for good reasons!) is equally important; when designing and implementing consistency maintainers, such mismatches between users' expectations and tool characteristics should be made explicit and discussed with all stakeholders. In this paper, we designed and performed an experiment asking 65 computer science students to answer questions regarding the expected behaviour of a synchroniser. Our contribution is twofold: (i) we propose an experiment design that can be adapted and repeated for a specific group of users, and (ii) we present and discuss empirical data obtained from two runs of the experiment.

Keywords bidirectional transformations; experiment; empirical studies; educational aspects of MBE.

1 Introduction and Motivation

A bidirectional transformation (bx) is a means of maintaining the consistency of two or more related sources of information [CFH+09]. Research on bx cuts across multiple domains including software engineering, graph transformation, programming languages, and databases. In this paper, we shall take terminology from the domain of Model-Driven Engineering (MDE) and refer to each related source of information as

Rebekka Wohlrab, Anthony Anjorin, Arjya Shankar Mishra. *What do Users Expect of Bidirectional Transformations?*. In Journal of Object Technology, vol. 19, no. 2, 2020, pages 4:1–19. doi:10.5381/jot.2020.19.2.a4

a *model* that is a member of some *model space* [DXC11a]. We shall also restrict our discussion to the simplified case of *binary* bx involving a consistency relation over two related model spaces. Developing a bx is challenging as consistency maintenance can involve multiple *operations* including checking for consistency, restoring consistency to one model after changing the other related model, and generating consistent pairs of connected models. Developing and maintaining these different operations as separate programs in a standard programming language is typically tedious and error-prone; the required programs tend to be similar enough to feel redundant to a programmer, yet are often still sufficiently different to defeat any naïve attempts at extracting common parts. Changing all operations consistently to avoid contradictions is thus an intricate task [Ste18]. To simplify bx development, diverse bx languages and tools have been proposed over the years. We refer to Anjorin et al. [ABW$^+$20] for a recent overview and benchmark results for a collection of bx tools.

As it is conceptually difficult to precisely capture the desirable behaviour of consistency maintainers, a considerable amount of work has been done to develop bx frameworks that establish basic *bx laws* governing the behaviour of "good" bx [ABW$^+$20]. The lens framework – in diverse variants – is especially well-accepted as a unifying formal framework for bx research [DXC11a, JR16]. In many cases, however, it is still unclear if such laws are too restrictive in practice or not, and if they really capture "desirable" behaviour from a user's perspective. Bidirectional transformations need to behave reasonably and as expected by their users, but researchers need a better understanding of users' expectations and intuitions [Ste10]. Indeed, previous work has found that it is still difficult to develop bx that end users are actually satisfied with [AYL$^+$18]. In addition to conceptual, methodical, technical, and tool-related challenges, an observation we have made in various projects is that end users are still not satisfied with solutions that bx experts might readily accept as "well-behaved". Based on (informal) discussions, collected feedback from project partners, and related work [AYL$^+$18, Ste10], our conjecture is that users have intuitive expectations that are formed and changed over the years by using various ubiquitous tools that perform some form of consistency maintenance. Examples include git, Dropbox, Google Docs, and numerous mobile apps for collaborative project management and social networking. While users' expectations can (arguably) be unreasonable, we claim that it is still beneficial to understand what most target users will expect from consistency maintainers for at least three reasons: (i) such expectations can serve as input for designing, extending, and improving existing bx frameworks and tools, (ii) knowing relevant users' expectations can help to choose the most suitable bx tool for a specific application scenario, and (iii) being aware of intuitive expectations that *cannot* be fulfilled by a chosen bx tool (perhaps for good reasons!) is important as such mismatches should be made explicit and discussed as early as possible with all stakeholders.

In this paper, therefore, we attempt to elicit and explore users' expectations of bx. We designed and performed an experiment asking 65 graduate computer science students to answer questions regarding the expected behaviour of a synchroniser. Our contribution is twofold: (i) we propose an experiment design that can be adapted and repeated for a group of users in the context of a specific project or application scenario, (ii) we present and discuss empirical data obtained from two runs of the experiment.

The rest of the paper is structured as follows: Section 2 presents a concrete example that was also used for the experiment. Section 3 provides an overview of the expectations we chose to explore with the experiment. Section 4 covers our research method and the format of the questions posed. Section 5 presents the results of the

experiment and discusses our interpretation of the empirical data obtained. Section 6 compares our contribution with related work, while Section 7 concludes with an outlook on possible future work.

2 An Interactive BX Demonstrator for the Experiment

As a brief introduction to bx for the reader, we present in this section the "bx demonstrator" used as an interactive, concrete example for the experiment. An overview of our bx demonstrator is depicted in Fig. 1. Explanatory videos on the bx demonstrator can be found online[1]. We designed the bx demonstrator in a way that it fulfills certain properties of consistency relations, as we explain in detail in Section 3. We posed the questions in the experiment in an example-independent way, but selected a domain that was generally known to users lacking deeper knowledge of bx. The example is familiar to a wide audience and as general as possible regarding the underlying consistency relation (i.e., not obviously a bijection). Inspired by kitchen planners, the bx demonstrator[2] is a web-based application providing two views on a virtual planning space: the left view is a *layout grid* ① divided into squares, while the right view is a symbolic view of the actual *kitchen* ② containing certain items (sinks, refrigerators, chairs, tables, etc.). Each of these items occupies a number of squares in the grid, arranged in a certain pattern. After performing changes to one of the views,[3] the user can press the *sync* button ③ to propagate these changes to the other view and restore consistency. Both views are consistent if suitable space has been reserved in the layout grid for every item in the kitchen. "Suitable space" is defined by a set of rules, stating, for example, that two neighbouring horizontal or vertical squares ④ are required for every sink ⑤ (one for the sink, one for enough space so the sink can be used). This means that extra squares (such as the reddish-brown square in the second row and fourth column) can be "blocked" in the layout grid (for windows, doors, lamps) without having a corresponding item in the kitchen.

Both views, however, also enforce certain constraints. The layout grid prevents an overlapping of coloured squares. Each colour represents a group of squares that belong together, and thus must be created and deleted together. Groups cannot be moved in the layout grid. The kitchen view, in contrast, allows items to be created without checking if there is actually sufficient space for them. Changes made in the kitchen, therefore, can be rejected when synchronising if they cannot be propagated consistently to the layout grid. The kitchen enforces two of its own constraints: the vertical left blue border represents a wall with water outlets, while the horizontal top red border represents a wall with electrical outlets. Certain items (such as sinks), can only be created close enough to the blue wall, while others (such as fridges) can only be created close enough to the red wall. As this is not checked in the layout grid, groups created for such items can be rejected when "forward" propagating to the kitchen. Finally, the demonstrator can interact with users to make decisions. For example, tables can be placed either horizontally or vertically, as long as there is sufficient space in the layout grid. In such cases, a pop-up dialogue ⑥ allows the user to choose a propagation strategy.

The example-independent notation used to pose our questions is adapted from

[1] `https://rebrand.ly/bxexperimentvideos`

[2] `https://github.com/ArjyaSMishra/BXToolDemonstrator`

[3] We focus on the simplified case of model synchronisation and leave model *integration* for future work, as allowing concurrent changes leads to a wide range of topics related to conflict resolution.

[ABW+20, AYL+18, DXC11a]. Figure 2 shows how the notation is connected to our concrete example. We consider a consistency relation over two *model spaces*, each consisting of all possible *models* in the space, and the changes (*deltas*) required to transition from one model to another. The demonstrator involves a layout grid model space and a kitchen model space. As depicted in the centre of Fig. 2, the two views of the demonstrator represent a pair of connected models, each in their corresponding model space. For example, the layout grid represents some model a in the model space A. When the user makes changes in one of the views, e.g., deleting and creating a group of blocks in the layout grid, these changes represent a delta $da : a \rightarrow a'$ going from the model a to another model a' representing the final state of the layout grid. To restore consistency, pressing the *sync* button triggers a *forward synchronisation*, which maps the input delta da to a suitable delta db in the output model space B (the kitchen model space). As depicted to the right of Fig. 2, the output delta $db : b \rightarrow b'$ is expected to be between models b and b' that correspond to models a and a', respectively. While this correspondence is implicit in the demonstrator, our notation uses *correspondence* links c and c' to make the relation between source and target models explicit. To simplify terminology in the rest of the paper, we refer to one of the model spaces (depicted to the left) as the *source*, and to the other model space (depicted to the right) as the *target* model space. *Forward* synchronisation, therefore, takes as input both a source delta da as well as a correspondence c and produces an output target delta db and a correspondence c'.

3 Hypotheses for the Experiment

The goal of this study was to understand users' expectations of consistency maintainers. Based on a review of existing bx frameworks [FGM+07, Ste10, Dis08, XSHT13, DXC+11b, Sch94, JR16], we considered three groups of expectations: expectations concerning the *consistency relation* to be maintained, expectations concerning the *domain of definition* for forward synchronisation, and expectations concerning the *behaviour* of forward synchronisation. These expectations reflect what bx researchers currently *think* users expect, as reflected in the formal properties suggested in the

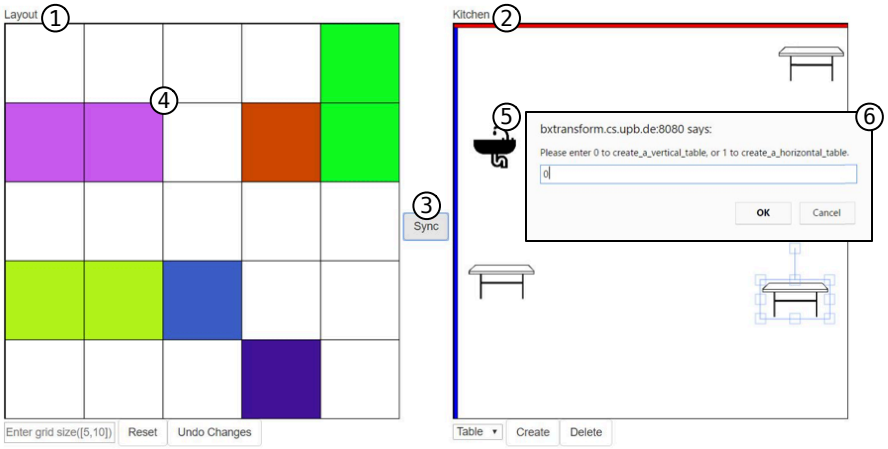

Figure 1 – A bx demonstrator used as part of the experiment

various bx frameworks. Investigating the three groups of expectations allows us to reflect on how well existing bx frameworks meet actual users' expectations.

For each group we characterise the relevant properties, provide an example based on the bx demonstrator, and formulate a suitable null hypothesis to be tested by the experiment. While we only discuss *forward* synchronisation in the following, backward synchronisation can be handled analogously.

3.1 Expectations Concerning the Consistency Relation

Two basic properties of a consistency relation are *totality* and *uniqueness* [ACG+14]:

Totality: A consistency relation over two model spaces (source and target) is *source total* if for every source model, there exists at least one corresponding target model such that the relation is satisfied. Target totality is defined analogously.

Uniqueness: A consistency relation over two model spaces (source and target) is *target unique* if for every source model, there exists at most one corresponding target model such that the relation is satisfied. Source uniqueness is defined analogously.

Example: The bx demonstrator is designed to be general in the sense that the underlying consistency relation is neither source nor target total; there exist both source and target models for which there can be no consistent target/source models due to the constraints imposed in each model space. Similarly, the underlying consistency relation is neither source nor target unique; there exist source and target models for which there are multiple consistent target/source models. This is due to the fact that multiple items can occupy the same space in the grid, and different grid patterns can be used to "place" the same item in the grid (horizontal or vertical tables).

As we suspect that many users tend to derive their intuitive expectations of bx from bijections, our hypothesis is that users have a clear expectation of totality and uniqueness. We thus formulate the following null hypotheses, which we shall attempt to reject with our experiment:

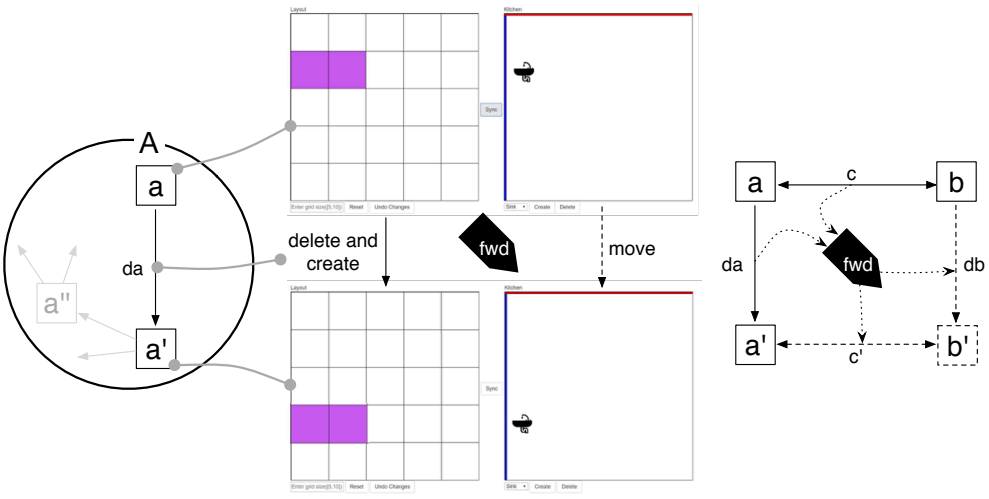

Figure 2 – Connection between formal notation and concrete example

(H1$_0$) *Users do not expect consistency relations to be necessarily source total.*

(H2$_0$) *Users do not expect consistency relations to be necessarily target unique.*

3.2 Expectations Concerning the Domain of the Forward Synchroniser

A forward synchroniser is typically expected to be "correct", i.e., to determine a corresponding target model such that consistency is restored (cf. Fig. 2). More controversial is what a forward synchroniser should take as input and depend on, i.e., what constitutes the domain of definition for the forward synchroniser. Possible candidates include the previous source and target models, the changed source model, the input source delta that effected this change, and the correspondence between the previous source and target models (as indicated in Fig. 2).

Example: Forward and backward synchronisation in the demonstrator depend on the correspondence between source and target models. Which groups of squares belong to which items influences how deletion and movement are propagated. As the grid layout only supports creation and deletion of groups of squares with session-unique colours, source deltas can be reconstructed from old and new source models. In contrast, it is in general impossible to distinguish movement of items from deletion and recreation, without inspecting the target delta. As a consequence, forward synchronisation does not require source deltas as input, while backward synchronisation does.

Due to the simplicity, predominance, and success of state-based tools and bx frameworks (especially state-based lenses), our conjecture is that most users' expectations are probably state-based (which also fits well with the expectation of a bijection). We thus formulate the following null hypotheses:

(H3$_0$) *Users have no expectations regarding the dependency of forward synchronisation on correspondence links.*

(H4$_0$) *Users have no expectations regarding the dependency of forward synchronisation on input deltas.*

3.3 Properties of the Forward Synchroniser

Numerous laws have proposed in an attempt to formally capture the *well-behavedness* of forward synchronisation. We refer to, e.g., Hidaka et al. [HTCH16] for an exhaustive overview. For our experiment, we focus on undoability as a basic law, which we expect to be intuitively understandable:

Undoability: Given consistent source and target models, a forward synchroniser is undoable [Ste10] if, after applying the forward synchroniser to propagate a source delta δ_S, it is always possible to (i) revert to the original source model (effectively *undoing* δ_S) via a source delta $undo(\delta_S)$, and (ii) obtain the exact same corresponding, original target model by propagating $undo(\delta_S)$.

Example: The bx demonstrator is not undoable in general. As an example, random but session-unique colours are assigned to the groups of squares in the grid layout. There is thus no guaranteed way of undoing changes so that the exact same colour scheme as before is obtained. As another example, deleting and recreating (undoing the deletion of) an item that requires some decision making will again require decision making – the user is then free to select a different option than before.

As all (text) editors and IDEs we are aware of support some form of undo and redo functionality, we assume that most users take undoability for granted and would be surprised if this were not supported. While bx researchers know that guaranteeing undoability in a bx setting poses strong limitations [Ste10], we doubt if this is obvious for the average software developer. As null hypothesis, therefore, we state that undoability is irrelevant with respect to users' expectations:

($\mathbf{H5}_0$) *Users have no clear expectations regarding undoability.*

4 Research Method

We conducted an experiment following Wohlin et al.'s guidelines [WRH+12]. The objective of the experiment was to understand intuitions and expectations of computer science students with respect to bx. To elicit general expectations that are not tied to a particular example, we used a visual notation (cf. Sect. 2) to represent generic situations without requiring a detailed understanding of formal terminology. To ensure that students understood the questions, the experiment was conducted in three steps: (1) an initial test was conducted to elicit expectations, (2) the participants then interacted with the bx demonstrator presented in Sect. 2, and (3) the same test was repeated.

We conducted two runs, Exp1 and Exp2, of the experiment. Exp1 was conducted with 40 computer science students in a course on Software Quality Assurance, Exp2 with 25 computer science students in a course on Model-Based Software Development. The students were not completely randomised as both courses are elective and are selected by students typically according to interest and convenience. The participants of Exp1 are from a larger spectrum of computer science students, including students with little experience and interest in software development, while most participants of Exp2 have a primary focus on software engineering with experience using developer tools such as git and a wide range of IDEs.

In the experiment, all students were asked to work through an initial tutorial[1] introducing the notation used for posing questions as well as very basic concepts (models, deltas, model spaces, synchronisation). The participants then took the first test including demographic questions (age, background, prior experience) and five (Exp1)/four (Exp2) questions regarding expectations connected to bx (cf. Sect. 3). The questions on bx expectations were multiple choice questions and included a Likert-scale question "how sure are you about your answer?", to record the participants' subjective confidence in their answers. Afterwards, students were asked to work with the bx demonstrator. In the second test, all participants worked through a few slides connecting the notation used for the questions with the demonstrator as an example. The same five (Exp1)/four (Exp2) questions regarding bx expectations were then repeated as in the initial test. In Exp1, one participant did not take the second test and was filtered out.

Figures 3 and 4 depict questions related to target uniqueness and source totality (cf. Sect. 3) from Exp1. Only two alternatives were provided, essentially forcing all participants to make a binary choice. Even though students' subjective confidence was recorded with a subsequent question, we decided to improve Exp2 by designing the questions to allow for simple sanity checks. We also added an additional option "I have no idea" with which subjects could explicitly indicate that they did not know the answer. The question from Exp2 related to source totality and target uniqueness

is depicted in Fig. 5. In this case, for an answer to be counted as "the participants expects the underlying consistency relation to be source total", options C, D, E, and F must *not* be checked. The questions for the tests and descriptions of the experiments are available online.[4] The participants were not aware of the actual hypotheses, but informed that the results would be used for research on bx. Confidentiality and anonymity was ensured and it was stressed that participation was voluntary.

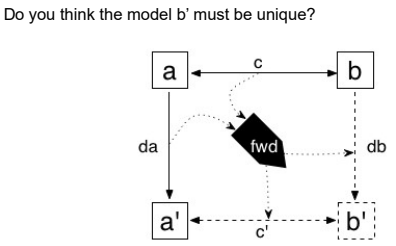

Figure 3 – Exp1: Question 1

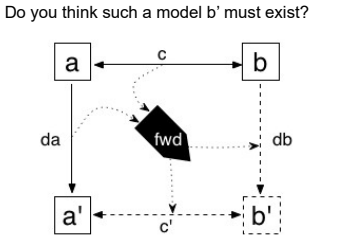

Figure 4 – Exp1: Question 2

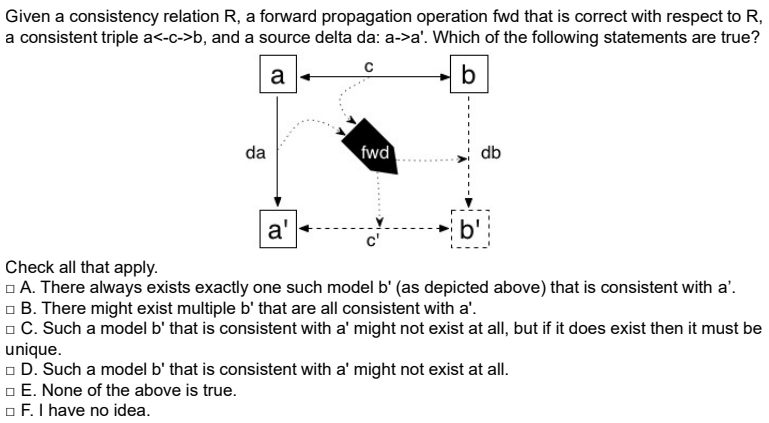

Figure 5 – Exp2: Question 1

4.1 Threats to Validity

We now discuss the most important threats to the validity of our experiment, together with resulting limitations and mitigation strategies we applied in each case.

Internal validity relates to confounding factors influencing the relationship between variables, treatment, and our results, especially with respect to causality. A major threat is that participants' answers are more influenced by the lack of understanding of the questions and notation used, and not by their real intuition and expectations. To mitigate this threat, we use a visual notation based on a few basic concepts, avoiding potentially confusing formal terminology. Furthermore, participants are asked to express their confidence in each answer. We use these values to weight answers and provide a more realistic picture. As confidence values are, however, subjective and might not be related to true understanding, we provide a tutorial introducing the notation used, and an interactive demonstrator allowing participants to map the questions to a concrete, practical example. As this specific example can in turn heavily

[4] `http://rebrand.ly/bxexperimentquestions`

influence answers, we conduct a first test *before* providing the demonstrator, and a second test *after* interaction with the demonstrator.

Construct validity relates to the extent to which hypotheses or theories for the experiment are actually tested with the measurements used. We aim to study students' expectations and intuitions with respect to bx and approached the topic from the perspectives of consistency relations, domain of definition, and behaviour of forward synchronisation. A threat here is that our scope was limited to aspects covered in the experiment; a more realistic picture requires further questions covering many more aspects, properties, and laws. We had to restrict our experiments, however, to four/five question to avoid fatigue. Moreover, users' expectations might be different when other examples are used than the kitchen planning example in our bx demonstrator. Further experiments are required to extend our current scope. A part of our experiment and results was example-independent. This part can be reused and compared to other results if the experiment is repeated for different examples. Another problem involves hypothesis guessing or evaluation apprehension, i.e., students striving to present themselves in a favourable light or in a manner that pleases the lecturer. To mitigate this threat, we communicated clearly that participation is voluntary, and that we ensure confidentiality and anonymity. We also refrained from stating concrete hypotheses and only mentioned that the goal of the experiments was to understand participants' expectations regarding bx. Finally, while subjective confidence values are useful, it is questionable if these ordinal values can be averaged as if they were interval values. We have taken care when interpreting these confidence values, and only use them as an additional source of information regarding participants' (subjective) understanding of the notation used.

Conclusion validity relates to missing correlations where they exist, or arriving at correlations or results that do not exist in reality. In this study, conclusion validity is compromised by the comparatively low number of samples. Our samples are also neither completely random nor completely homogeneous. The subjects in the experiments come from different backgrounds and years of study, and participated in experiments with two setups and sets of questions (Exp1 and Exp2), further strengthening this threat. As mitigation, we were careful to avoid data dredging and made sure to report all contradictory results that were not in line with our initial hypotheses.

External validity relates to the generalisability of our results. Our samples are relatively small, and are potentially not representative for all computer science students, let alone all users. The use of students instead of industrial practitioners has been criticized by empirical researchers in the past, but using students as participants has been found a valid simplification [FJW+18], especially when studying technologies unknown to the participants [SMJ15]. While we tried to mitigate threats to external validity by involving participants from different courses at different stages of their studies, it is clear that our results only hold for similar populations: Exp1 for users with a broad IT-related background, and Exp2 for software developers. Further experiments are required to refute or support our results, which should be regarded only as a starting point.

5 Results and Discussion

In this section, we cover the expectations proposed in Sect. 3. In each case, we present the data obtained from both runs of the experiment, discuss if the null hypothesis – users are ambivalent and have no clear expectation – can be rejected, and provide our

interpretation of the results. All raw and derived data is available online.[5]

5.1 Source Totality

Results concerning source totality are depicted in Table 1. All values are provided in percentage to simplify comparison across experiments. Every entry in the table is of the form *X(x) / Y(y)*, where X is the value obtained for an expectation (X% of all users expect totality) and Y for its negation (Y% of all users do not). The values in parentheses are determined by averaging the corresponding subjective confidence values (normalised to a number between 0 and 1). While the X and Y values always add up to 100% for Exp1, this is not the case for Exp2 due to sanity checks and filtering of self-contradictory answers. The data is arranged in two columns: the left column for results before interacting with the demonstrator and the right column for results after, and in two rows: the first row for Exp1, the second for Exp2. While percentages are easier to compare, statistical significance depends on the size of the sample in each case. To take this into account we calculate corresponding p-values for each entry using a binomial test with $p = 0.5$. This represents attempting to reject the null hypothesis, i.e., that users are equally likely to expect source totality as not. Taking significance at the 5% level, we indicate that an entry in the table is significant, i.e., p-value < 0.05, by making it bold. In such cases, the null hypothesis can be rejected and an actual expectation is thus likely.

In both experiments and in all cases *more participants appear to expect source totality*. This conclusion can be made with statistical significance for Exp1, before and after interaction with the demonstrator. For Exp2, the conclusion is only significant for the second test. In both experiments and in all cases, participants became subjectively more confident of their answers (on average) after interacting with the demonstrator as a concrete example. We believe this indicates that the demonstrator helped improve (subjective) understanding of the notation used and content of the questions. In Exp2, more subjects were able to answer the questions without self-contradictions after interacting with the demonstrator (76% vs. 84%). Important is also that the interaction does not appear to have changed the general expectation of source totality. Finally, in most cases the average confidence in Exp2 is slightly higher than in Exp1. This can be perhaps explained by our design — in Exp2 we are able to perform sanity checks and filter out self-contradictory answers.

Discussion: While the expectation of source totality corroborates our practical experience, it is surprising that the demonstrator served to strengthen instead of weaken this tendency. The demonstrator is after all *not* source total and shows that some input deltas must be rejected. Free-text comments indicate that some students might have assumed such cases to be bugs in the demonstrator.

If the indicated expectation of source totality does hold for most users, what could be an explanation for this? We believe that users intuitively expect to be able to

Table 1 – Expectation of source totality: Yes / No in %

	before (average confidence)	after (average confidence)
Exp1 (n = 40)	**62.5** (0.56) / **37.5** (0.61)	**66.7** (0.76) / **33.3** (0.72)
Exp2 (n = 25)	40.0 (0.65) / 36.0 (0.72)	**60.0** (0.73) / **24.0** (0.79)

[5] `http://rebrand.ly/bxexperimentdata`

propagate changes that they were allowed to make. Indeed, after spending time and effort to effect such a change, it is frustrating if pressing "sync" does not work. Bx researchers know that fulfilling this expectation is challenging: the change API in the source domain has to somehow take the entire bx into account and magically allow only exactly the changes that can be forward propagated successfully. We propose instead a different consequence for bx research: while model synchronisation (propagating changes from one model to the other) is easier to implement (in theory and practice), model *integration* with support for conflict resolution is inevitable. Our conjecture is that current software engineers – probably influenced by git and Dropbox – expect to be able to *always* propagate their changes, even if this leads to conflicts, branching and merging (which have to be well-supported of course). Further experiments could explore this further using semi-structured interviews to try to find out *why* source totality is preferred.

5.2 Target Uniqueness

Results for expectations of target uniqueness are depicted in Table 2 (analogously to Table 1) for both experiments. Before interaction with the demonstrator, *most participants appear not to expect target uniqueness*, i.e., seem ready to accept possibly multiple consistent results when forward synchronising. Before interacting with the demonstrator, this conclusion is significant for both Exp1 and Exp2. After interaction with the demonstrator, the tendency is strengthened for Exp2 but weakened for Exp1 to the point where the null hypothesis can no longer be rejected. In all cases, interacting with the demonstrator increased subjective confidence and the total number of valid answers in Exp2.

Discussion: As the demonstrator allows the user to choose from multiple options, the effect on Exp2 makes sense; the effect on Exp1, however, does not. A possible explanation is that the demonstrator does not actually *produce* multiple results but involves the user in the process of making the synchronisation unique. This might not have been obvious for the participants, especially non-programmers in Exp1.

The apparent acceptance of non-uniqueness corroborates our practical experience, especially in an industrial context [AYL+18], where we found it typically unacceptable to fix all details of a bx to the point where synchronisation is unique. Our conjecture is that users probably do not mind – or maybe even expect – to have an *interactive* system that can ask for (perhaps optional) additional information to complete its tasks.

Table 2 – Expectation of target uniqueness: Yes / No in %

	before (average confidence)	after (average confidence)
Exp1 (n = 40)	**12.5** (0.52) / **87.5** (0.53)	51.3 (0.81) / 48.7 (0.80)
Exp2 (n = 25)	**16.0** (0.50) / **60.0** (0.73)	**16.0** (0.81) / **72.0** (0.74)

5.3 Dependency on Correspondence Links

Results concerning correspondence links are depicted in Table 3. While all entries show that *participants expect synchronisation results to depend on correspondence links*, this tendency only becomes significant after interacting with the demonstrator.

An increase in subjective confidence and the number of valid answers for Exp2 can be observed after interaction with the demonstrator.

Discussion: Most subjects appear to think that the correspondence between models plays an important role for synchronisation. We expect the demonstrator to strengthen this tendency as there is a clear, albeit implicit connection between groups of squares of a certain colour in the grid, and the corresponding item in the kitchen. Without this information it would be completely unclear what is to be synchronised, and how this is to be done correctly.

While it might be obvious enough that how things correspond is important for synchronisation, our questions (unfortunately) do not clarify if subjects understand that this correspondence cannot always be computed and must be provided as an input. This would be an interesting insight to be explored with future experiments. As users appear to regard correspondence links as important, a possible consequence for the design of synchronisation tools would be to make correspondence links as explorable and explicit as possible.

Table 3 – Dependency on input correspondence links: Yes / No in %

	before (average confidence)	after (average confidence)
Exp1 (n = 40)	57.5 (0.50) / 42.5 (0.74)	**69.2** (0.60) / **30.8** (0.75)
Exp2 (n = 25)	44.0 (0.50) / 28.0 (0.64)	**60.0** (0.63) / **20.0** (0.70)

5.4 Dependency on Input Deltas

Results concerning dependency of forward synchronisation on input deltas are depicted in Table 4. While a slight tendency can be observed towards the expectation that *input deltas do play a role for synchronisation*, none of the entries is significant. Interaction with the demonstrator serves to weaken this tendency further, even though average confidence and the number of valid answers for Exp2 still increase.

Discussion: Many tools we use as software developers (still) work with text and are parser- and state-based. It typically does not make much of a difference how exactly you perform a change; only the final text is taken into account when you save (and "synchronise"). This is perhaps reflected in our results that show no clear expectation. While the demonstrator is actually delta-based and records the exact changes performed, this might not be obvious enough to the participants. Considering the low number of valid answers in Exp2 (only 32% before, and 36% after interaction with the demonstrator), our results indicate that this is perhaps still a confusing question that should be rephrased and improved in the experiment. An alternative explanation, however, could be that the majority of software developers do not necessarily think in terms of deltas and more easily understand states (models). A consequence for the design of delta-based synchronisers is that special care must be taken to present and explain explicit input deltas to users as this can otherwise be a primary source of confusion.

Table 4 – Dependency on input deltas: Yes / No in %

	before (average confidence)	after (average confidence)
Exp1 (n = 40)	60.0 (0.53) / 40.0 (0.45)	51.3 (0.72) / 48.7 (0.74)
Exp2 (n = 25)	20.0 (0.50) / 12.0 (0.67)	20.0 (0.60) / 16.0 (0.69)

5.5 Undoability

Table 5 depicts our results regarding undoability of forward synchronisation. Most subjects appear to *expect synchronisation to be undoable*. This conclusion is significant in all cases. While interaction with the demonstrator increased average confidence as usual, the tendency was strengthened in Exp1 but weakened in Exp2.

Discussion: Most users probably take an undo button for granted in software applications and tools. In this regard, interacting with a demonstrator that is *not* undoable does not seem to have made a substantial difference. While undoability is regarded as being desirable (and our results corroborate this), it is also a strong limitation for bx [Ste10] and can thus be a problematic expectation or requirement. As a consequence for the design and implementation of synchronisers, therefore, the fact that synchronisation will probably *not* be undoable in general should be discussed and clarified with potential users early enough in the development process.

Table 5 – Expectation of undoability: Yes / No in %

	before (average confidence)	after (average confidence)
Exp1 (n = 40)	**67.5** (0.61) / **32.5** (0.52)	**71.8** (0.79) / **28.2** (0.71)
Exp2 (n = 25)	**56.0** (0.70) / **16.0** (0.63)	**52.0** (0.83) / **20.0** (0.80)

6 Related Work

The idea for this paper and the proposed experiment is partly inspired by existing work [TOW+17, OM18] on transferring the extensive body of research on *organisational justice* (see e.g., [CZ15] for a review) to technical domains such as human-computer interaction. Organisational justice is concerned with how and why people experience "fairness" in their working environment. As experiments indicate that many if not most results can be transferred to the case where one of the agents of interaction is a software system [OM18], research on organisational justice can serve as a source of inspiration for designing synchronisation and consistency management tools. Indeed, such a transfer of ideas has been suggested by Töniges et al. [TOW+17], who apply strategies for achieving organisational justice to the design of adaptive cyber-physical systems. The goal is to ensure that human agents feel fairly treated when collaborating with (robotic) agents completely controlled by software. While our current results only motivate and set the stage for future empirical research and experiments, promising hypotheses can be derived from how users would feel fairly treated by synchronisation tools, especially in the advanced case of model integration and unavoidable conflict resolution.

Several controlled experiments have been suggested or conducted to study users' expectations of transformation languages [ASS18, KHK+16, HSB+18]. A design template for a planned experiment was proposed to compare model transformation languages to general-purpose languages, with a focus on understandability [KHK+16]. In a controlled experiment using Xtend, ATL and QVT-O, several tasks were found to be challenging for users, even for well-qualified subjects [HSB+18]. No statistically significant evidence could be found that domain-specific transformation languages lead to a better performance than general-purpose languages [HSB+18]. While these existing empirical studies provide insights into how end users learn and apply transformation languages, we are not aware of any studies focusing on bidirectional transformations in particular.

To the best of our knowledge, none of the existing bx frameworks [FGM+07, Ste10, Dis08, XSHT13, DXC+11b, Sch94, JR16] are based on any form of empirical evidence concerning what most users would characterise as "desirable" behaviour. While the proposed laws are often argued convincingly and are certainly based on the substantial experience of the respective authors, the current generation of active software developers today might have very different preferences, e.g., expecting highly interactive, configurable tools as opposed to fully automated systems.

Existing bx frameworks also tend to have a distinct domain-specific "flavour": frameworks [FGM+07, KZH16] from the programming language community are typically state-based, making the assumption that the input delta can be ignored by synchronisers. This is, however, probably less related to what users expect and more connected with current parsing technology and what kinds of parser-based tools can be more easily realised. There is no denying that state-based formulations are simply easier to understand: initial work by Stevens and Diskin [Ste10, Dis08, XSHT13] probably chose a state-based setting as a simplification and a point to start from (and not as a preferred setting). Indeed, Diskin goes on to suggest a delta-lens framework [DXC+11b], with numerous arguments and examples for why input deltas (and explicit correspondence links) should be considered a primary source of information for synchronisers.

In contrast to a parser-based setting with textual artefacts, the *model-driven* setting of the Triple Graph Grammar (TGG) approach [Sch94, ALS15] is typically naturally delta-based, assuming that an explicit account of changes made is readily available. This is again probably specific to graph-based, MDE tool building, as it is completely unclear if users prefer working with and being aware of deltas; one could argue that input deltas, as a representation of changes to be propagated, are much too fine grained for most users, who do not care what exactly was changed to result in the final model. As a grammar-based approach, TGGs are naturally non-unique, i.e., relational and not functional, allowing multiple consistent target (source) models for the same source (target) model. While this seems to be appreciated in practice [AYL+18], with most bx tools allowing for flexible runtime configuration [ABW+20], it is unclear how much flexibility should be provided. Indeed, some bx approaches [MC19] allow the bx program that defines consistency to be itself manipulated as part of the consistency restoration process. While this might be considered too flexible by most bx researchers, it has led to very successful bx tooling.

Finally, it is interesting to observe a mismatch between certain theoretical frameworks and their practical realisations. As an example, BiGUL [KZH16] is based on state-based, *total* lenses but is partial in practice. This is again probably due to the practical difficulty of testing all possible input for validity, as opposed to simply executing the synchroniser and concluding this from failure.

7 Conclusion and Future Work

In this paper, we explored users' expectations of bx. We addressed the challenge of doing this without requiring a deep understanding of formal terminology by using a visual notation to describe relevant situations, and to pose questions in an example-independent manner. To ensure that the questions were understood, we prepared an interactive bx demonstrator and used it to provide a connection between the generic notation and a concrete example. We recorded answers to all questions before and after interacting with the demonstrator, and also collected subjective confidence values.

Our results from two runs (Exp1 and Exp2) of the experiment show that the demonstrator increased average subjective confidence, reduced the number of self-contradictory answers (we could check this in Exp2), and did not change the general tendency in most cases. We thus argue that our experiment design achieves an acceptable compromise between using a specific example to improve understanding, but still posing questions in a generic manner to elicit general, example-independent expectations.

As it is impossible to make any definitive conclusions based on just two runs of the experiment and a limited number of participants (in total 65 computer science students), our results can only provide initial indications that must be supported or refuted by future experiments:

- Users appear to expect source totality, showing surprise and irritation when a synchroniser rejects input. Considering that many bx tools do *not* guarantee source-totality [KZH16, ALS15], this suggests that conflict resolution in the context of model integration is an important and still largely open challenge for bx research.
- Users do not appear to mind non-unique results, and seem ready to interact and cooperate with synchronisers to determine the final output. In this context, the open challenge for bx research is to embrace non-uniqueness and explore *how* best to interact with users without overwhelming them with low-level details. Simply enumerating potentially hundreds of results, for example, probably won't be acceptable for users.
- While many users appear to understand the importance of correspondence links, it might be more challenging for users to understand the influence of input deltas on synchronisation as opposed to just providing the changed model as input. As bx researchers such as Diskin et al. argue for delta-based bx tooling [DXC+11b], an open challenge here is to understand better how to design synchronisers that allow users to view, inspect, and manipulate explicit deltas as input and output (as opposed to implicit, hidden delta creation).
- We fear that most users probably expect synchronisation steps to be undoable. While an undo/redo "button" seems to be a basic requirement, it is already clear that it is severely limiting for synchronisation tools [Ste10]. In the context of model integration, taking interaction and conflict resolution into account will only make it even less realistic to expect undoability as a formal guarantee. The consequence here, therefore, is more for bx requirements engineering: undoability with all consequences should be discussed and clarified as early as possible, together with potential alternatives such as fine-grained versioning.

As future work, similar experiments based on our design could be repeated for different groups of users varying age, background, gender, and professional experience. Having developed and tested the experimental design in this study constitutes an important step in understanding what target users expect from bx frameworks and tools. One direction for future work is to conduct the experiment with industrial practitioners and bx experts. Our current catalogue of questions barely covers very basic expectations – further experiments can extend this to cover expectations related to model integration, conflict resolution, and other bx laws. Our results could be complemented with experiments that focus on qualitative aspects, using for example semi-structured interviews to find out if users can *explain* their expectations and preferences.

References

[ABW+20] Anthony Anjorin, Thomas Buchmann, Bernhard Westfechtel, Zinovy Diskin, Hsiang-Shang Ko, Romina Eramo, Georg Hinkel, Leila Samimi-Dehkordi, and Albert Zündorf. Benchmarking bidirectional transformations: theory, implementation, application, and assessment. *Software and Systems Modeling*, 19:647–691, 2020. doi:10.1007/s10270-019-00752-x.

[ACG+14] Anthony Anjorin, Alcino Cunha, Holger Giese, Frank Hermann, Arend Rensink, and Andy Schürr. Benchmarx. In *Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014)*, pages 82–86, 2014. URL: http://ceur-ws.org/Vol-1133/paper-13.pdf.

[ALS15] Anthony Anjorin, Erhan Leblebici, and Andy Schürr. 20 years of triple graph grammars: A roadmap for future research. *Electronic Communications of the EASST*, 73, 2015. doi:10.14279/tuj.eceasst.73.1031.

[ASS18] Vlad Acreţoaie, Harald Störrle, and Daniel Strüber. VMTL: a language for end-user model transformation. *Software and Systems Modeling*, 17(4):1139–1167, 2018. doi:10.1007/s10270-016-0546-9.

[AYL+18] Anthony Anjorin, Enes Yigitbas, Erhan Leblebici, Andy Schürr, Marius Lauder, and Martin Witte. Description languages for consistency management scenarios based on examples from the industry automation domain. *Programming Journal*, 2(3):7, 2018. doi:10.22152/programming-journal.org/2018/2/7.

[CFH+09] Krzysztof Czarnecki, J. Nathan Foster, Zhenjiang Hu, Ralf Lämmel, Andy Schürr, and James F. Terwilliger. Bidirectional transformations: A cross-discipline perspective. In *Proceedings of the Second International Conference on Theory and Practice of Model Transformations (ICMT 2009)*, pages 260–283, 2009. doi:10.1007/978-3-642-02408-5_19.

[CZ15] Jason A. Colquitt and Kate P. Zipay. Justice, fairness, and employee reactions. *Annual Review of Organizational Psychology and Organizational Behavior*, 2(1):75–99, 2015. doi:10.1146/annurev-orgpsych-032414-111457.

[Dis08] Zinovy Diskin. Algebraic models for bidirectional model synchronization. In *Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2008)*, pages 21–36, 2008. doi:10.1007/978-3-540-87875-9_2.

[DXC11a] Zinovy Diskin, Yingfei Xiong, and Krzysztof Czarnecki. From state- to delta-based bidirectional model transformations: the asymmetric case. *Journal of Object Technology*, 10:6: 1–25, 2011. doi:10.5381/jot.2011.10.1.a6.

[DXC+11b] Zinovy Diskin, Yingfei Xiong, Krzysztof Czarnecki, Hartmut Ehrig, Frank Hermann, and Fernando Orejas. From state- to delta-based

bidirectional model transformations: The symmetric case. In *Proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2011)*, pages 304–318, 2011. doi:10.1007/978-3-642-24485-8_22.

[FGM+07] J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems*, 29(3):17, 2007. doi:10.1145/1232420.1232424.

[FJW+18] Davide Falessi, Natalia Juristo, Claes Wohlin, Burak Turhan, Jürgen Münch, Andreas Jedlitschka, and Markku Oivo. Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Software Engineering*, 23(1):452–489, Feb 2018. doi:10.1007/s10664-017-9523-3.

[HSB+18] Regina Hebig, Christoph Seidl, Thorsten Berger, John Kook Pedersen, and Andrzej Wasowski. Model transformation languages under a magnifying glass: a controlled experiment with Xtend, ATL, and QVT. In *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, 2018*, pages 445–455. ACM, 2018. doi:10.1145/3236024.3236046.

[HTCH16] Soichiro Hidaka, Massimo Tisi, Jordi Cabot, and Zhenjiang Hu. Feature-based classification of bidirectional transformation approaches. *Software and Systems Modeling*, 15(3):907–928, 2016. doi:10.1007/s10270-014-0450-0.

[JR16] Michael Johnson and Robert D. Rosebrugh. Unifying set-based, delta-based and edit-based lenses. In *Proceedings of the 5th International Workshop on Bidirectional Transformations (Bx 2016)*, pages 1–13, 2016. URL: http://ceur-ws.org/Vol-1571/paper_13.pdf.

[KHK+16] Max E. Kramer, Georg Hinkel, Heiko Klare, Michael Langhammer, and Erik Burger. A controlled experiment template for evaluating the understandability of model transformation languages. In *Proceedings of the Second International Workshop on Human Factors in Modeling co-located with ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016)*, volume 1805, pages 11–18, 2016.

[KZH16] Hsiang-Shang Ko, Tao Zan, and Zhenjiang Hu. BiGUL: a formally verified core language for putback-based bidirectional programming. In *Proceedings of the 2016 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM 2016)*, pages 61–72, 2016. doi:10.1145/2847538.2847544.

[MC19] Mikaël Mayer and Ravi Chugh. A bidirectional Krivine evaluator. In *Proceedings of the 8th International Workshop on Bidirectional Transformations (Bx@PLW 2019)*, pages 56–60, 2019. URL: http://ceur-ws.org/Vol-2355/paper5.pdf.

[OM18] Sonja K. Ötting and Günter W. Maier. The importance of procedural justice in human–machine interactions: Intelligent systems as new decision agents in organizations. *Computers in Human Behavior*, 89:27–39, 2018. doi:https://doi.org/10.1016/j.chb.2018.07.022.

[Sch94] Andy Schürr. Specification of graph translators with triple graph grammars. In *Proceedings of the 20th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 1994)*, pages 151–163, 1994. doi:10.1007/3-540-59071-4_45.

[SMJ15] Iflaah Salman, Ayse Tosun Misirli, and Natalia Juristo. Are students representatives of professionals in software engineering experiments? In *Proceedings of the 2015 IEEE/ACM IEEE International Conference on Software Engineering (ICSE'15)*, pages 666–676. IEEE, 2015. doi:10.1109/ICSE.2015.82.

[Ste10] Perdita Stevens. Bidirectional model transformations in QVT: semantic issues and open questions. *Software and Systems Modeling*, 9(1):7–20, 2010. doi:10.1007/s10270-008-0109-9.

[Ste18] Perdita Stevens. Is bidirectionality important? In *Proceedings of the 14th European Conference on Modelling Foundations and Applications (ECMFA 2018)*, pages 1–11, 2018. doi:10.1007/978-3-319-92997-2_1.

[TOW+17] T. Töniges, S. K. Ötting, B. Wrede, G. W. Maier, and G. Sagerer. Chapter 26 - an emerging decision authority: Adaptive cyber-physical system design for fair human-machine interaction and decision processes. In Houbing Song, Danda B. Rawat, Sabina Jeschke, and Christian Brecher, editors, *Cyber-Physical Systems*, Intelligent Data-Centric Systems, pages 419–430. Academic Press, Boston, 2017. doi:https://doi.org/10.1016/B978-0-12-803801-7.00026-2.

[WRH+12] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer, Berlin, Heidelberg, 2012. doi:10.1007/978-3-642-29044-2.

[XSHT13] Yingfei Xiong, Hui Song, Zhenjiang Hu, and Masato Takeichi. Synchronizing concurrent model updates based on bidirectional transformation. *Software and Systems Modeling*, 12(1):89–104, 2013. doi:10.1007/s10270-010-0187-3.

About the authors

Rebekka Wohlrab is currently with Systemite AB, Gothenburg, Sweden. She holds a PhD from Chalmers University of Technology. Contact her at rebekka.wohlrab@systemite.se, or visit https://www.chalmers.se/en/staff/Pages/wohlrab.aspx.

Anthony Anjorin is currently a Junior Professor for Model-Based Software Development at Paderborn University. Contact him at anthony.anjorin@upb.de, or visit https://anthonyanjorin.github.io.

Arjya Shankar Mishra is currently a Software Developer at Fastec GmbH, Paderborn, Germany. Contact him at `arjya.sm@gmail.com`, or visit `https://www.linkedin.com/in/arjyashankarmishra/`.

Acknowledgments This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The authors would like to thank Klementina Josifovska for her support during the development of the study.