The Families to Persons Case — Revisited

Anthony Anjorin[1,†], Thomas Buchmann[2,**,†]

[1]Zühlke engineering GmbH, Hannover, Germany
[2]Hof University of Applied Sciences, Hof, Germany

Abstract

This paper revisits the Families to Persons Case [1] with a significant extension: concurrent model synchronization. Building on the original benchmark test cases, we introduce new tests for synchronizing models and resolving conflicts, thereby enhancing the framework's capability to benchmark bidirectional transformation tools under more realistic conditions. This advancement is crucial for assessing the tools' performance in concurrent engineering scenarios requiring data consistency across multiple models.

Keywords

Model transformations, model management, concurrent model synchronization, bidirectional transformations, benchmarks

1. Introduction

The *Families to Persons* (F2P) case has long served as a foundational benchmark in the bidirectional transformations (bx) community. Originating from the ATL transformation zoo, it was later integrated into the Benchmarx framework [2] to enable systematic evaluation of bx tools. The case centers on a simple yet expressive model-to-model transformation between a family registry and a person register, with well-defined consistency relations and a suite of forward and backward transformation tasks. Over the years, it has been used to assess the correctness, expressiveness, and performance of a wide range of bx tools [1, 3].

However, while the original F2P case effectively exercises basic bidirectional transformation capabilities, it largely omits one of the most challenging and practically relevant aspects of bx: model synchronization under concurrent updates. In real-world modeling scenarios, such as collaborative modeling, model merging, or distributed development, models often evolve independently on both sides, requiring robust synchronization mechanisms that can detect, handle, and resolve conflicts.

This paper presents a revised and significantly extended version of the F2P case, specifically designed as a call for solutions in the domain of consistent model synchronization (csync). Rather than evaluating a fixed set of tools, our goal is to invite participants to apply their own bx or synchronization approaches and report how they address the challenges posed by concurrent, potentially conflicting edits.

The extended case study retains all original forward and backward transformation tasks to ensure backward compatibility and baseline validation. In addition, it introduces 13 new synchronization test cases that cover a spectrum of concurrent model changes, including both conflict-free scenarios and realistic conflict patterns such as move-delete, rename-rename, and delete-rename conflicts. These cases are carefully designed to probe the capabilities of csync mechanisms in terms of correctness, conflict handling, and scalability.

We do not prescribe any particular technology or architecture. Instead, we encourage submissions from a wide range of paradigms: declarative, operational, graph-based, text-based, or hybrid. We welcome both automated and interactive conflict resolution strategies. By collecting and comparing diverse solutions, we aim to stimulate discussion, reveal new insights, and ultimately advance the state of the art in model synchronization.

TTC 2026
*Corresponding author.
†These authors contributed equally.
anthony.anjorin@zuehlke.com (A. Anjorin); thomas.buchmann@hof-university.de (T. Buchmann)
0000-0001-6213-6243 (A. Anjorin); 0000-0002-5675-6339 (T. Buchmann)

The paper is structured as follows: Section 2.1 outlines the original F2P case and its consistency constraints. Section 2.2 details the new synchronization scenarios and conflict types. Section **??** details the model synchronisation case, while Section 4 defines the expected outcomes, including correctness, conflict handling, and scalability measurements. The remaining sections present the technical setup, execution guidelines, and a template for solution submissions.

2. Case Description

This case revisits the well-known *Families to Persons* (F2P) transformation, a classic bidirectional transformation example from the ATL Zoo, as a benchmark for evaluating tools supporting *concurrent synchronisation (csync)*. While prior work using the Benchmarx framework [3] focused on one-sided synchronisation (where changes on one model are propagated to the other), this year's TTC edition advances the challenge by requiring support for *simultaneous, bidirectional updates*.

2.1. Original Case

We present an extended F2P benchmark within *Benchmarx 2.0* [4], a generalised framework that supports csync: given consistent models and concurrent edits on both sides, a bx tool must restore consistency by synchronising both models in a single step. One-sided scenarios are treated as special cases (one idle delta).

The F2P case models two organisational registers — families and persons — over the same population (see Figure 1). Consistency requires:

- Structural conformance to respective metamodels.
- Exactly one register per model.
- A bijection between family members and persons, preserving gender and ensuring person names follow the format "familyName, memberName".

2.2. Extension for concurrent edits

Changes are expressed as explicit edit models, enabling fine-grained inspection and control. Figure 2 illustrates a test case where Homer is deleted and Ned created in the family model, while Homer and Marge are deleted in the person model. A correct solution must propagate Ned's creation, confirm Homer's consistent removal, and delete Marge in the family model to preserve requested changes.

The benchmark includes a `concurrent` test suite divided into (see Figure 3):

- `MonotonicCreating` / `MonotonicDeleting`: simultaneous creations or deletions,
- `NonMonotonic`: mixed edits,
- `Conflicts`: competing or inconsistent changes.

Solutions are evaluated not only on pass/fail outcomes but on how they handle csync semantics, including conflict resolution and edit preservation. The case has been implemented using three csync approaches: propagation-based (*BXtend*), grammar-based (*IBeX*), and hybrid (*eMoflon::Neo*), demonstrating the benchmark's adaptability.

This TTC case invites participants to implement the F2P transformation with support for concurrent synchronisation, offering a practical and scalable challenge grounded in real-world data consistency problems.

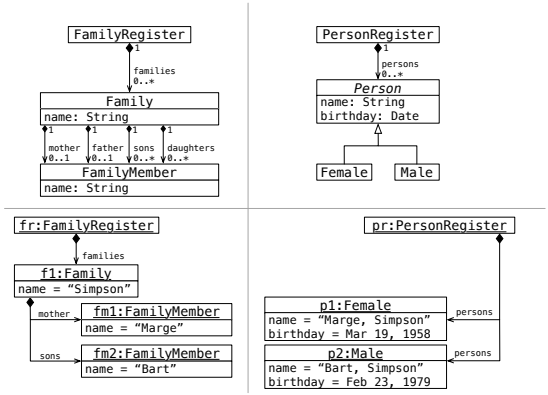

Figure 1: Families to persons (F2P) metamodels and consistent models

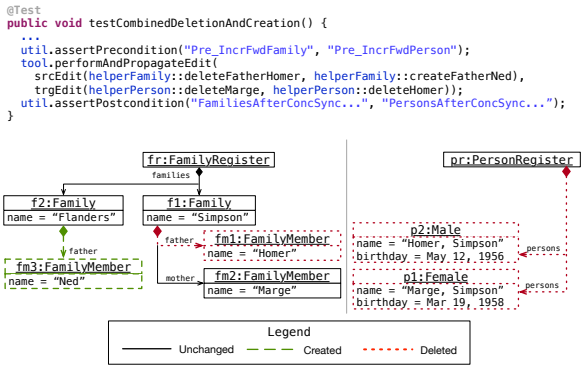

Figure 2: Concurrent edit example: deletion and creation in both models

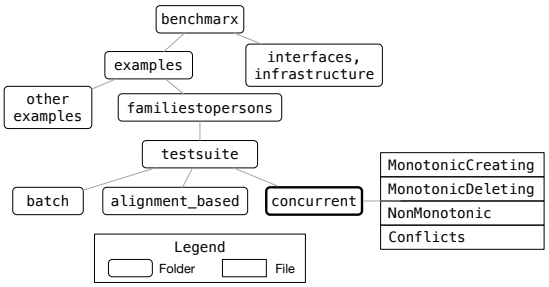

Figure 3: Structure of the F2P Benchmarx

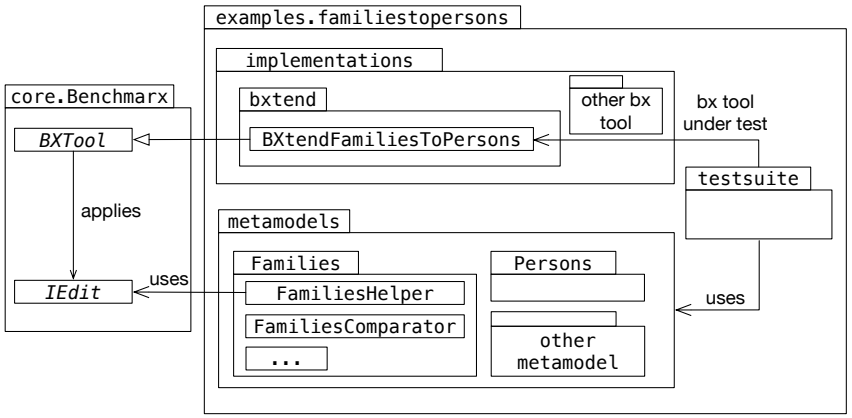

Figure 4: Overview of the Benchmarx framework applied to the Families-to-Persons (F2P) benchmark. The architecture includes metamodels, model helpers, normalizers, comparators, and the extended interface for concurrent synchronization.

3. Model Synchronization

This section presents the evolution of the Benchmarx benchmarking framework into Benchmarx 2.0, specifically extended to support concurrent model synchronization—a scenario where edits occur independently and simultaneously on both sides of a bidirectional transformation. We detail key design decisions, interface extensions, and the application of these enhancements in the context of the Families-to-Persons (F2P) benchmark. An overview of the framework architecture is illustrated in Figure 4.

3.1. Extending the Bx Tool Interface for Concurrency

To support concurrent synchronization, the core interface between test cases and bidirectional tools has been significantly extended. The primary method, `performAndPropagateEdit`, now accepts two edits—one for each model direction—to be processed in a single synchronization step.

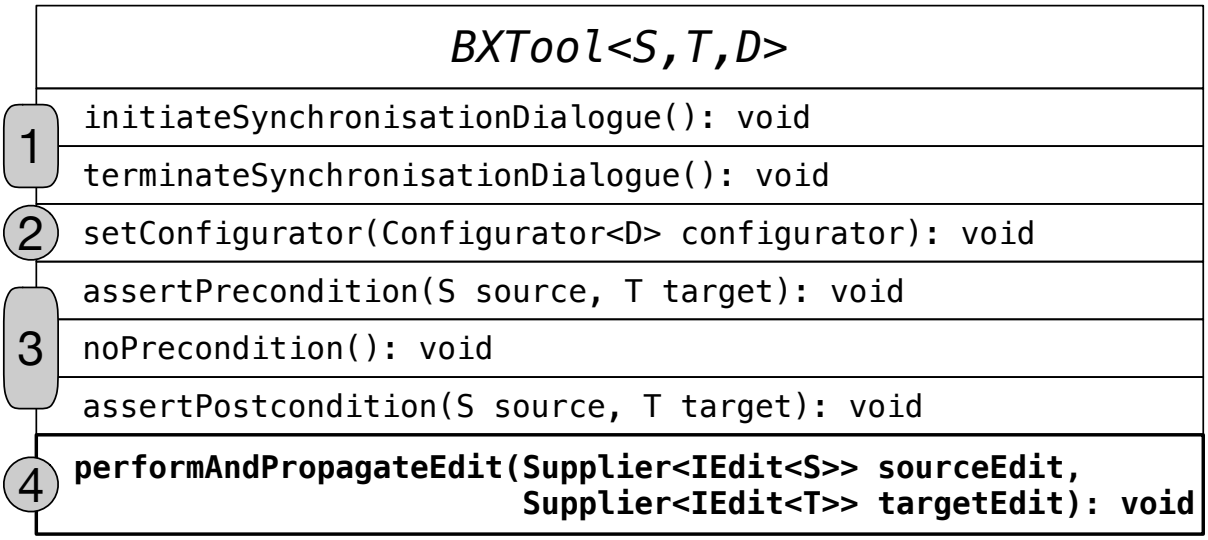

Figure 5: Extended Bx tool interface supporting concurrent edit propagation. New elements are highlighted to reflect bidirectional input and flexible result handling.

As shown in Figure 5, the updated interface includes the following key components:

- **Precondition assertion (③)**: Verifies that the system reaches the expected initial state after setup.
- **Concurrent edit execution (④)**: The central method `performAndPropagateEdit` now takes suppliers for both a source and a target edit (`() -> IEdit`), allowing the tool to decide when to instantiate them—enabling integration with change listeners or lazy evaluation strategies.
- **Flexible result semantics**: Unlike one-sided synchronization, where a single correct outcome is typically expected, concurrent edits may admit multiple valid consistent states. To accommodate this, Benchmarx 2.0 supports several pragmatic evaluation strategies:

1. **Unique preferred outcome**: The test case designer asserts that, despite theoretical alternatives, one result is clearly most reasonable based on criteria such as edit preservation or minimal divergence.
2. **Finite set of acceptable outcomes**: A predefined set of consistent end states is provided; the tool passes if it produces any member of this set.

3. **Configurable behavior via parameters**: The test case supplies configuration options through the `Configurator` (see Figure 4), enabling the tool to adapt its strategy dynamically (e.g., favoring retention vs. rejection of edits). The space of valid parameters must be clearly documented in the benchmark specification.

4. **Objective function with threshold**: For complex scenarios, the test case may implement an example-specific metric (e.g., structural similarity, edit fidelity) and define a pass/fail threshold. This offers maximum flexibility at the cost of increased implementation effort.

These strategies reflect the inherent ambiguity in defining "correctness" in concurrent model synchronization [5], and allow benchmark authors to balance precision with practicality.

3.2. Supporting Structures: Helpers, Normalizers, and Comparators

To facilitate robust and reproducible testing, Benchmarx integrates auxiliary components (see Figure 4):

Model helpers: Provide reusable edit patterns (e.g., create family, delete person) to simplify test case construction. **Normalizers and comparators**: Handle structural variations (e.g., element ordering, attribute defaults) that are irrelevant to consistency, ensuring fair comparison of model states.

These utilities reduce boilerplate code and help isolate behavioral differences due to synchronization logic rather than syntactic discrepancies.

3.3. Concurrent Synchronization in the F2P Benchmark

We applied these extensions to the Families-to-Persons (F2P) benchmark[1], adding 13 new test cases focused on concurrent synchronization. A representative example, `testCombinedDeletionAndCreation`, is illustrated in Figure 2.

The test begins by applying a setup sequence (not shown) to establish the precondition, which is then verified using a utility function that delegates comparison to the bx tool. Following this, a concurrent edit is issued:

In the *family register*, `Homer` is deleted and `Ned` is created. In the *person register*, both `Homer` and `Marge` are deleted.

To restore consistency:

- `Homer`'s deletion is already consistent across models—no action needed.
- `Marge` is deleted only in the person register. The postcondition requires her deletion to be propagated to the family register, aligning with the principle of preserving user-intended edits where possible [6].
- The creation of `Ned` in the family register should be mirrored by creating a corresponding male person with a default birthday, as specified by the benchmark.

This case exemplifies a non-monotonic edit pattern involving both creation and deletion operations.

3.4. Categorization of Concurrent Test Cases

The 13 new test cases are categorized as shown in Table 1, covering a spectrum of synchronization challenges.

We distinguish between:

- **Monotonic** cases (only deletions or only creations), which test basic propagation under independence.

[1] https://github.com/eMoflon/benchmarx/tree/main/examples/familiestopersons

Table 1
New F2P test cases for concurrent synchronisation

Category	Description	#
Monotonic Deleting	Non-conflicting deletions in both models.	3
Monotonic Creating	Non-conflicting creations in both models.	4
Non-Monotonic	A mix of non-conflicting deletion and creation in both models.	2
Conflicts	Move/delete conflict	1
	Rename/rename conflict	1
	Delete/rename conflict	1
	Move/rename	1

- **Non-monotonic** cases, combining opposing operations and challenging the tool's ability to resolve interdependent changes.
- **Conflict** scenarios, where edits interfere semantically.

Given the structure of the F2P metamodels:

- Conflicts arise from operations such as moving a family member (changing surname), renaming (first name), or deleting individuals.
- In the person register, renaming may affect both first and last names, potentially conflicting with family-level edits. Since persons are unordered and lack explicit role annotations, certain conflicts (e.g., move/create, delete/create, create/rename) do not manifest.
- Notably, even apparent create/create conflicts (e.g., adding `Marge` as mother in `Simpson` family and as person `Marge, Simpson`) are resolvable: the person could be synchronized as a daughter instead, avoiding inconsistency due to role multiplicity constraints.

Thus, the conflict set is carefully curated to reflect only those situations that genuinely challenge consistency maintenance.

4. Intended Results and Evaluation Goals

In addition to verifying the correctness of forward and backward transformations—based on the original Families-to-Persons (F2P) benchmark [2], our primary objective is to evaluate how bidirectional tools handle *concurrent model synchronization (csync)*. We have extended the benchmark with 13 new test cases (Section 3.3) that involve simultaneous edits on both models, including both *conflict-free* and *conflicting* scenarios.

We expect solution providers to:

- Correctly synchronize all concurrent edits while preserving consistency.
- Explicitly describe their tool's *conflict detection and resolution strategy*, particularly for semantic conflicts such as move/delete, rename/rename, and delete/rename.
- Clarify whether conflicts are resolved automatically, interactively, or avoided via operational restrictions.

A key goal is to assess not only functional correctness but also the *pragmatic viability* of csync approaches in realistic collaborative modeling environments where concurrent changes are common.

4.1. Runtime Scalability Analysis

As part of this case study, we require solution providers to perform scalability measurements using their respective tools and report the results. Experiments should be conducted under consistent conditions, and each data point must be based on the median of at least five independent runs to ensure stability. Participants are expected to detail their experimental setup, including hardware, software environment, and any tool-specific configurations, to enable reproducibility. The following four experiments are to be executed for each supported csync approach, with results reported separately for each tool.

Forward and Backward Batch Transformations We measure the time to create a consistent target model from scratch (forward: families $\rightarrow$ persons; backward: persons $\rightarrow$ families) for models of increasing size. **Model size**: n families with 5 members each (forward); $5n$ people grouped by surname (backward).

RQ1: Is a csync approach efficient enough for batch transformations, or should simpler, dedicated batch mechanisms be preferred?

Forward and Backward Incremental Transformations We apply a small edit (e.g., add one family member or person) to one model and measure synchronization time, given an initially consistent pair of models of size n.

RQ2: Can csync tools achieve perfect incrementality—i.e., constant-time updates independent of model size?

Csync with Fixed Edits, Increasing Model Size We perform concurrent edits on models of growing size (n families with 5 members and corresponding persons):

- *Conflict-free*: Add a son to a family + delete an unrelated female person.
- *Conflict-inducing*: Move a daughter between families + delete her in the person register.

Each pair is repeated three times per configuration.

RQ3: Are csync updates incremental in practice? How does performance scale with model size, and how significantly does conflict resolution impact runtime?

Csync with Fixed Model, Increasing Edit Size Using a fixed model of 500 families (2,500 people), we vary the number n of concurrently edited families, applying the same two edit patterns.

RQ4: How does synchronization time depend on edit size? Does conflict resolution lead to non-linear overhead?

These experiments allow us to assess not only correctness but also the *runtime efficiency and scalability* of csync tools under realistic usage patterns—key factors for adoption in collaborative modeling environments.

5. Conclusion

In this paper, we propose an extension to a previous TTC Case — The Families to Persons Case [1]. By focusing on *concurrent model* synchronisation in this TTC Case, we address an important generalisation of one-sided model synchronisation from the original case.

The Benchmarx framework has been extended to support the new test cases, which are designed to evaluate the correctness and efficiency of bidirectional transformation tools in synchronising models and resolving conflicts. The new test cases are based on the original Families to Persons Case, but

they introduce additional requirements and constraints to test the tools' capabilities in more complex scenarios.

The case can be found in the Benchmarx Github Repository: https://github.com/eMoflon/benchmarx In the examples subfolder a folder "familiestopersons-ttc2026" contains all required models and test cases. It also contains a JAR file with the reference implementation and example code for integrating the respective solutions in the benchmarx runner.

References

[1] A. Anjorin, T. Buchmann, B. Westfechtel, The families to persons case, in: A. García-Domínguez, G. Hinkel, F. Krikava (Eds.), Proceedings of the 10th Transformation Tool Contest (TTC 2017), co-located with the 2017 Software Technologies: Applications and Foundations (STAF 2017), Marburg, Germany, July 21, 2017, volume 2026 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017, pp. 27–34. URL: https://ceur-ws.org/Vol-2026/paper2.pdf.

[2] A. Anjorin, Z. Diskin, F. Jouault, H.-S. Ko, E. Leblebici, B. Westfechtel, Benchmarx reloaded: A practical framework for bidirectional transformations, in: R. Eramo, M. Johnson (Eds.), Sixth International Workshop on Bidirectional Transformations (BX 2017), CEUR Workshop Proceedings, 2017.

[3] A. Anjorin, T. Buchmann, B. Westfechtel, Z. Diskin, H. Ko, R. Eramo, G. Hinkel, L. Samimi-Dehkordi, A. Zündorf, Benchmarking bidirectional transformations: theory, implementation, application, and assessment, Softw. Syst. Model. 19 (2020) 647–691. URL: https://doi.org/10.1007/s10270-019-00752-x. doi:10.1007/S10270-019-00752-X.

[4] A. Anjorin, T. Buchmann, L. Fritsche, Benchmarx 2.0: A benchmark for concurrent model synchronisation approaches, in: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, 2024, pp. 950–959.

[5] J. Cheney, J. Gibbons, J. McKinna, P. Stevens, On principles of least change and least surprise for bidirectional transformations, J. Object Technol. 16 (2017) 3:1–31. URL: https://doi.org/10.5381/jot.2017.16.1.a3. doi:10.5381/JOT.2017.16.1.A3.

[6] F. Orejas, A. Boronat, H. Ehrig, F. Hermann, H. Schölzel, On propagation-based concurrent model synchronization, Electron. Commun. Eur. Assoc. Softw. Sci. Technol. 57 (2013). URL: https://doi.org/10.14279/tuj.eceasst.57.871. doi:10.14279/TUJ.ECEASST.57.871.