Towards a Repository of Bx Examples

Towards a Repository of Bx Examples

James Cheney, James McKinna, Perdita Stevens
School of Informatics
University of Edinburgh
firstname.lastname@ed.ac.uk

Jeremy Gibbons
Department of Computer Science
University of Oxford
firstname.lastname@cs.ox.ac.uk

ABSTRACT

We argue for the creation of a curated repository of examples of bidirectional transformations (bx). In particular, such a resource may support research on bx, especially cross-fertilisation between the different communities involved. We have initiated a bx repository, which is introduced in this paper. We discuss our design decisions and their rationale, and illustrate them using the now classic *Composers* example. We discuss the difficulties that this undertaking may face, and comment on how they may be overcome.

1. INTRODUCTION

Research into bidirectional transformations (henceforth: **bx**) involves a wide range of disciplines, from databases, to model-driven development, to programming languages. The literature is rich in examples illustrating the corresponding wealth of notations and tools used to formalise the variety of bx as they occur "in the wild".

In writing a paper about bidirectional transformations, one typically needs to use examples. Several examples, such as the notorious UML class diagram to RDBMS schema example, have appeared in many variants in papers by many authors. It can be difficult to identify whether examples in different papers are really identical; moreover, the need to fully define every example mentioned often makes it difficult to use more than one or two examples in a paper that is subject to a page limit, and can lead to examples being defined so concisely as to make it difficult for a reader to understand them. This, in turn, leads to proliferation of versions.

In 2013 at the ETAPS Bx workshop[1], we announced the development of a repository of examples, to be made publicly available on the Bx wiki [14]. We have a personal interest in such a repository, as a foundation for our own work in the EPSRC-funded project *A Theory of Least Change for Bidirectional Transformations*; but we hope that it will have much broader use. The repository aims to simplify reuse of examples, especially in order that meaningful comparisons between formalisms will be easier to make. It aims to improve communication between sometimes quite disparate communities, and to help explain bx to interested outsiders. We hope that researchers in bx will both add their examples to the repository, and refer to examples from the repository as appropriate. This should have benefits both for authors and for readers. Naturally, papers will still have to be sufficiently self-contained; but we think that the repository may still be helpful. For example, if one's paper illustrates some new work using an existing example, one might describe the example briefly (as at present), focusing on the aspects most relevant to the work at hand. One can, however, also give a reference into the repository, where there is more detail and discussion of the example, which some readers may find helpful. Over time, we might expect that certain examples in the repository become familiar to most researchers in bx. Then, if a paper says it uses such an example, the reader's task is eased. The repository should also be a good place to link in auxiliary materials, such as files showing how an example is implemented in different formalisms, or diagrams suitable for inclusion in papers and talks.

Repositories of examples are not easy to make successful, however; if this one is to succeed, it is important to design its formats and processes with care, and to engage the community. This paper describes our initial ideas and the rationale behind them.

Concretely, we illustrate what we have in mind through the *Composers* example. We do not intend that every example in the repository will follow the format illustrated here – different examples may have different needs. Our intention is that our template will at least provide a basis for discussion for future iterations. The repository of examples is hosted on the Bx wiki at:
`http://bx-community.wikidot.com/examples:home`

2. DESIGN AND RATIONALE

We draw some inspiration from the design patterns movement [6], but not so much in the sense of 'a catalogue of expert advice for assisting novices'. Indeed, there are socio-technical arguments as to why this is a rather difficult case to make [5]. An easier case to make for the benefits of patterns is that they improve communication between already proficient practitioners; and it is this analogy that we hope to make in considering the design of the repository.

One way in which design patterns improve communication is simply by capturing and recording 'folk knowledge' – the concepts and stories known and shared by everyone in a community, and assumed to be so known, so not always explained in detail. By explicitly recording our corpus of common examples, we remind ourselves of their existence, and we provide a resource for filling in the inevitable gaps in our coverage of this 'folk knowledge'.

A second way in which design patterns improve communication is by providing a common vocabulary. The naming of entries is a difficult matter; but a well-chosen name takes its place in the community's discourse, evoking a concept or a story and invoking its connotations. But for the purposes of scientific communication,

[1] `http://bx-community.wikidot.com/bx2013:home`

the name itself is not enough. We need also to be able to maintain a stable reference for each example in the repository, so that it can be referenced in a paper with some hope that that reference will persist. We also need to have some confidence that example descriptions themselves are stable; so we expect to have to curate the repository, encouraging free discussion and commentary but versioning the descriptions at appropriate points.

Also extrapolating from the history of design patterns, we suggest that there is value in having some regularity of the structure between examples in the repository. We propose a standard template for our examples, which we discuss in more detail below. We do not intend to be too prescriptive and rigid about this template; for one thing, we expect that our understanding of the best structure will evolve with experience, and for a second, examples arising from different parts of the bx community, or examples that differ in character, may well warrant different structures.

It should be noted that one way in which the example repository diverges from the patterns movement is that the entries in the repository do not follow Alexander's three-part rule of context, problem, and solution – there are no conflicting forces to resolve, nor necessarily any expectation of a resolution. Rather, if the examples are to be considered 'problems' at all, then it is in the Kuhnian sense of the problems and techniques that form a paradigm of 'normal science' [9]. Nevertheless, we intend that the repository will provide access to artefacts as well as descriptions, be they executable code, proof scripts, sample inputs and outputs, or what have you.

We wish to maintain a "broad church" approach to what can be included in the repository. We think that the most useful entries will be small ones that can be defined precisely, but in a way that is as independent as possible of any particular bx formalism. These we intend to describe principally in English (perhaps augmented with small amounts of simple mathematical notation) but with sufficient precision that readers familiar with a particular formalism should be able to tell, unambiguously, whether a representation of a bx in that language is correctly implementing the example. The review process, to be discussed, should play an important role in eliminating ambiguities. Artefacts showing how examples are implemented in particular formalisms may also be helpful.

Besides these, several other classes of examples may be anticipated. We agree with the authors of [1] (in this volume) that benchmarks may be seen as a distinct class and therefore should be included. Industrial-scale examples, accompanied by appropriate artefacts, would clearly be of interest, but equally clearly, could not be expected to be explained with full precision separately from their artefacts. Other examples we have in mind are more like sketches: situations in which a certain bx would clearly have applicability, but where details have not been worked out. These might be of particular benefit to outsiders wondering whether bx are of interest to them. We have adopted the suggestion made in group discussion at the Bx workshop in Banff in December 2013[2], namely to make explicit the class to which an example belongs, because these classes may be quite different in character.

3. A TEMPLATE FOR BX EXAMPLES

We propose the following template for examples. It consists of the following headings, whose kernel is the description of bx given, for example, by Stevens in [13]. That is, an example will typically define two or more classes of models, together with a consistency relation between them, and appropriate consistency restoration functions. Such functions might depend just on the states of the models (state-based bx) or might require as input extra information, e.g. concerning the edit that has been done. Either is fine provided it is clear what is assumed.

We propose the following standard fields and their order. Optional fields are indicated by '?' in the fieldname; other fields should be present, even if brief.

Title A descriptive name, such as COMPOSERS, by which authors may refer to the example. More advanced indexing, and traceability back to the originating sources (see under **References** below) may prove necessary as the repository grows.

Version 0.x for unreviewed examples.

Type For example, PRECISE, INDUSTRIAL, SKETCH. Use common sense concerning whether to use one or more: for example, PRECISE and SKETCH should be mutually exclusive, but conceivably either might be combined with INDUSTRIAL.

Overview A thumbnail description of the example, not more than two or three sentences; might include a brief summary of the example and/or a brief reason for its inclusion in the repository ("demonstrates [some interesting point]" for example).

Models Descriptions of the models, possibly with (formal) expressions of their meta-models. (We remind readers that we use the term "model", and "meta-model" inclusively: any appropriately precise description of the information sources being transformed is acceptable.)

Consistency Description of the consistency relationship between models. This should at least be in natural language, but may be augmented by formal expression in some language cognate with that of the meta-models.

Consistency Restoration Explain in which of the typically many possible ways inconsistencies are to be repaired. May be divided into separate descriptions of forward and backward restoration.

Properties? What additional properties are expected to hold of, or be exemplified by, the transformation? These will link to a separate glossary of terms such as 'hippocraticness'.

Variants? A difficult issue that we have found arises in writing examples is how to handle the choice points. Typically in making an example precise, even a small one, one realises that there are several points where more than one choice is reasonable. These multiply to give a potentially unmanageable set of examples. Our proposal is that one "base" example should be given in the main body of the text, and variation points be described here.

Discussion Origin, utility, interest, representativeness, related examples in the literature, ...

References? Bibliographic data for the paper or papers from which the example is taken, or where it is discussed, if applicable.

Authors Contributing author(s) of the example to the repository.

Reviewers? We intend that examples remain provisional (version 0.x) until reviewed (and approved, if necessary after modification) by other members of the wiki. In the interest of traceability and credit, such reviewers are identified here.

[2] http://www.birs.ca/events/2013/5-day-workshops/13w5115

Comments This is where any member of the wiki can comment. As discussed later, we do not wish to have uncontrolled editing of the example itself, but it is important to make it easy for community members to make comments. These might guide the development of a later version of the example.

Artefacts? Formal descriptions, perhaps downloadable code, sample input and output, virtual machine instances, diagrams...

4. AN EXAMPLE INSTANCE

We choose to illustrate our ideas with the familiar example of *Composers*, which first appeared in [3] and has been used by several others since. This version is closest to the one in [12].

Title COMPOSERS

Version 0.1

Type PRECISE

Overview This example stands for many cases where two slightly, but significantly, different representations of the same real world data are needed. The definition of consistency is easy, but there is a choice of ways to restore consistency.

Models A model $m \in M$ comprises a set of (unrelated) objects of class Composer, representing musical composers, each with a name, dates and nationality.

A model $n \in N$ is an ordered list of pairs, each comprising a name and a nationality.

Consistency Models m and n are consistent if they embody the same *set* of (name, nationality) pairs. That is, both: (i) for every composer in m, there is at least one entry in the list n with the same name and nationality; and (ii) for every entry in n, there is at least one element of m with the same name and nationality (there may be many such, each with distinct dates).

Consistency Restoration Given models m and n,

Forward produce a modified version of n by:

- deleting from n any entry for which there is no element of m with the same name and nationality;
- adding at the end of n an entry comprising each (name, nationality) pair derivable from an element of m but not already occurring in n.
  Such additional entries should be in alphabetical order by name, and within name, by nationality; no duplicates should be added (even if there are several composers in m with the same name and nationality).

Backward produce a modified version of m by:

- deleting from m any composer for which there is no entry in n with the same name and nationality;
- adding to m a new composer for each (name, nationality) pair that occurs in n but is not derivable from an element already occurring in m. The dates of any newly added composer should be ????-????.

Properties

- Correct
- Hippocratic
- Not undoable
- Simply matching

Variants If the bx is to be correct and hippocratic, restoring consistency must involve adding (respectively, deleting) composers that are present (respectively, not present) in the authoritative model but not in the one to be modified. Questions that the bx programmer still needs to resolve are:

- Do we ever modify the name and/or nationality of an existing composer, or do we create a new composer in the event of any mismatch? E.g. if one side has *Britten, British* and the other has *Britten, English*, does consistency restoration involve changing one of the nationalities, or adding a second Britten? Of course, if name is a key in the models then there is no choice.
- Where in the list n is a new composer added? Choices include: at the beginning; at the end. We might consider in an alphabetically determined position, but note that the user is not constrained to add composers in alphabetical order, and we fail hippocraticness if we choose to reorder when nothing at all need be changed. It therefore seems unlikely that changing the order of user-added composers will be wanted.
- What dates are used for a newly added composer in m?

Discussion This has been used as an example of why undoability is too strong. Consider a composer currently present (just once) in both of a consistent pair of models. If we delete it from n, and enforce consistency on m, the representation of the composer in m, including this composer's dates, is lost. If we now restore it to n and re-enforce consistency on m, then the absence of any extra information besides the models means that the dates cannot be restored, so m cannot return to exactly its original state.

References This version was used in:

Perdita Stevens, "A Landscape of Bidirectional Model Transformations", in *Generative and Transformational Techniques in Software Engineering II*, 2008, Springer LNCS 5235, pp408–424. DOI `10.1007/978-3-540-75209-7_1`

Original (asymmetric) variant was in:

Aaron Bohannon, J. Nathan Foster, Benjamin C. Pierce, Alexandre Pilkiewicz, and Alan Schmitt. "Boomerang: Resourceful Lenses for String Data". In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL)*, San Francisco, California, January 2008. DOI `10.1145/1328438.1328487`

Author(s) Perdita Stevens, James McKinna, James Cheney. University of Edinburgh.

Reviewer(s) None yet

Comments None yet

5. DIFFICULTIES AND HOW THEY MAY BE TACKLED

We are only too aware that it is far easier to found a repository such as proposed here than it is to ensure its success. In this section we attempt to address some questions that a sceptical reader might ask.

5.1 Can we ensure and maintain the quality of the repository contents?

A key decision we have made is that the example repository will be a *curated* resource in the sense of Buneman *et al.* [4]. This means that it is put together by human effort by members of a knowledgeable community – as opposed, say, to being extracted automatically from a web crawl. Other examples include many databases constructed to cover various areas of biomedical research (sometimes using wikis, sometimes using custom-developed websites). Example repositories in software engineering, such as ReMoDD [2], SPLOT [11], 101companies [10], and the Hillside patterns wiki [7], could also be viewed as curated in this sense; notice that there is a wide range of curation styles. One can also view Wikipedia as an example of this phenomenon, and some biological wikis are hosted there. These efforts have common technical and organizational structure (and encounter similar problems) and our aim is to learn from them.

Resources such as Wikipedia can be edited by anyone with an Internet connection. Unhelpful contributions are common, with editors and bots reverting such changes frequently. By contrast, resources curated by smaller, focused communities typically do not have this problem, especially if there is a barrier to entry, such as registration to obtain a wiki account. This is the model we plan to follow.

Specifically, we propose a three-level curatorial structure for the repository. Anyone with a wiki account will be able to comment on an example, and this should enable anybody to point out problems, ambiguities, extensions and any other points of interest. Eventually, each example will also have one or more named reviewers: recognised members of the community whose name as reviewer indicates they consider the example to be of usable quality. Overall editorial control of the repository is the responsibility of a small group of curators, initially ourselves.

At a technical level, there is a tension between fixing a structure for the curated data (such as our proposed example template) and allowing free-text or lightweight markup only. Both fixing a template and having no template at all can lead to headaches later; the former risks being too rigid for the application and forcing a premature decision about what the best template is, while the latter diminishes the integrity and intelligibility of the resource. We aim to take a middle road, providing a suggested template but not a barrier to varying it where good reasons to do so arise. In the longer term, adopting a more structured solution (e.g. to facilitate a move to a different platform than a wiki) may be justifiable.

5.2 Will anybody use it?

It has frequently been observed (*e.g.* by Josuttis in [8]) that companies that attempt to increase software reuse by creating a library of reusable components frequently find it easier to get people to put things into the library than to take them out.

In this case, informal discussion of the idea at two Bx meetings (in Rome and Banff) together with separate discussions with individuals suggest that there is a genuine need for examples and a willingness to refer to examples in a repository.

To help maximise the chance that the repository is used, we will seek to:

- host the repository on the main long-lived community site, the Bx wiki;
- use a format that makes it easy to find information quickly;
- provide guidance on how to refer to examples;
- keep old versions of examples available, so that old references can still be followed;
- introduce a reviewing mechanism to help ensure high quality of the examples.

To give more detail, we may split this question into several parts.

Will relevant people know about the repository?

Our aim in talking about our plans even before the repository existed was not only to check that we were targeting a genuine community need, but also to spread the word. The present paper is an attempt to give more detail. Further, by hosting the repository on the main Bx community site, the Bx wiki, we hope it will be easy to find.

Will people be able to find and refer to relevant examples?

Suppose that we have succeeded in answering the earlier question positively so that the repository does contain relevant, high quality examples: this is still no use unless people can find and use them. We might worry about a future in which there are so many examples in the repository that it is hard to find the right one; but experience suggests that ensuring that the wiki is (as at present) google indexed goes a long way to solving this problem.

In this specific case, another important aspect of use is the ability to refer accurately to an example, for example, in a paper that uses it. Readers seeing the reference need to be able to identify exactly the example referred to.

As already discussed, well-chosen names are important for the usability of our examples. However, versioning and variability management are likely to be problems; we have discussed this issue among ourselves and it has been raised by others at the Banff workshop also. It is important to distinguish between *versioning* and *variation*. Both require assigning distinct identifiers, but versioning connotes a (sequential) evolution of a single example, while variation connotes related variants of similar examples.

We could devise a system in which a short identifier represented the example, its version and the choice at each variation point. However, in the absence of automated support for such a system, we believe that taking care to choose example names carefully, to identify a single main variation within each example, and to maintain a linear sequence of numbered versions, will be more usable, and adequate for the time being.

Will anybody but ourselves put any examples in?

Experience with curated databases and crowdsourcing projects suggests that it is very important to pay attention to incentives for contributing (and maintaining) quality examples. We have already noted the importance of versioning; in addition to providing unique identifiers, it seems like a good idea to recommend a format for citations to examples (including versions) or to the repository itself. In addition, we hope that listing authors and reviewers of examples will incentivise contributions of examples and of effort to review them. If the repository reaches a point of relative maturity or stability, it may make sense to collect the most recent versions of all of the examples in it into a manuscript (with all authors and reviewers named), and publish it formally as a citable, archival technical report.

5.3 Will the contents rot?

For the next three years, we the authors can undertake to curate the repository under the aegis of the Least Change research project.

By that time, we hope that it will have gathered enough momentum that if necessary, finding other volunteer curators may be possible.

5.4 What happens if the Bx wiki dies?

We hope that the Bx wiki will flourish. We shall, however, maintain a local copy of the repository contents, in case of future difficulties. We plan to give some thought to whether maintaining it in a wiki-markup-independent form, and maintaining consistency between that and the wiki via a bidirectional transformation, might add value.

6. CONCLUSION

In the spirit of earlier documentary movements in the software engineering literature, such as that of the Patterns community, we have articulated the rationale for a repository of example bx, as well as providing here a canonical reference in the literature to its existence.

We hope that colleagues in the Bx community will wish to contribute to the repository, to use it in their own work and papers, and to discuss with us improved versions of the template described here. An early example is discussion that has just begun with authors of [1] about extra optional sections that may be necessary for benchmark examples.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge the contributions made by participants of the two 2013 Bx workshops, in Rome and Banff, and by the anonymous reviewers. The work reported here was supported by the EPSRC-funded project *A Theory of Least Change for Bidirectional Transformations* (EP/K020218/1 and EP/K020919/1).

8. REFERENCES

[1] Anthony Anjorin, Manuel Alcino Cunha, Holger Giese, Frank Hermann, Arend Rensink, and Andy Schürr. BenchmarX. In *Proceedings of Bx'14*, 2014.

[2] James M. Bieman, Betty H. C. Cheng, and Robert B. France. ReMoDD: The repository for model-driven development. `http://www.cs.colostate.edu/remodd/`.

[3] Aaron Bohannon, J. Nathan Foster, Benjamin C. Pierce, Alexandre Pilkiewicz, and Alan Schmitt. Boomerang: Resourceful lenses for string data. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), San Francisco, California*, January 2008.

[4] Peter Buneman, James Cheney, Wang-Chiew Tan, and Stijn Vansummeren. Curated databases. In *Proceedings of the 2008 Symposium on Principles of Database Systems (PODS 2008)*, pages 1–12, 2008. Invited paper.

[5] Luciana d'Adderio, Rick Dewar, Ashley Lloyd, and Perdita Stevens. Has the pattern emperor any clothes? A controversy in three acts. *ACM SIGSOFT Software Engineering Notes*, Jan/Feb 2002. `http://dx.doi.org/10.1145/566493.1148026`.

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[7] Hillside Group. The Hillside patterns wiki. `http://c2.com/cgi/wiki`.

[8] Nicolai M. Josuttis. *SOA in Practice: The Art of Distributed System Design*. O'Reilly, 2007. `http://www.soa-in-practice.com/`.

[9] Thomas S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, 1962.

[10] Ralf Lämmel. The 101companies project. `http://101companies.org/`.

[11] Marcilio Mendonca. SPLOT: Software product lines online tools. `http://www.splot-research.org/`.

[12] Perdita Stevens. A landscape of bidirectional model transformations. In Ralf Lämmel, Joost Visser, and João Saraiva, editors, *Generative and Transformational Techniques in Software Engineering (GTTSE)*, volume 5235 of *Lecture Notes in Computer Science*, pages 408–424. Springer, 2007.

[13] Perdita Stevens. Bidirectional model transformations in QVT: Semantic issues and open questions. *Journal of Software and Systems Modeling (SoSyM)*, 9(1):7–20, 2010.

[14] The Bx Community. Bidirectional transformations: The Bx wiki. `http://bx-community.wikidot.com/`.