A Demonstrator Framework for Consistency Management Approaches

by

Arjya Shankar Mishra

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

A Demonstrator Framework for Consistency Management Approaches

Master's Thesis

Submitted to the Fakultät für EIM, Institut für Informatik
in Partial Fulfillment of the Requirements for the
Degree of

Master of Science

by
Arjya Shankar Mishra

Supervisor:
Jun. Prof. Dr. Anthony Anjorin

Paderborn, April 1, 2017

Declaration

I hereby declare that I prepared this thesis entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them. This thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

City, Date

Signature

Contents

1 Introduction

Bidirectional transformation (denoted by "bx") is a technique used to synchronize two (or more) instance of different meta-models. Both models are related, but don't necessarily contain the same information. Changes in one model thus lead to changes in the other model [1]. Bx makes sure that two models that can change over time have to be kept constantly consistent with each other.

Bidirectional transformation is used to deal with scenarios like:

- change propagation to the user interface as a result of underlying data changes
- synchronization of business/software models
- refreshable data-cache incase of database changes
- consistency management between two artifacts by avoiding data loss

and many more....

1.1 Contribution

Bx community has been doing research and development work in many fields like software development, database, mathematics and much more to increase awareness and to reach more people [2][1]. As a result, many kinds of bx tools are being developed, e.g., eMoflon [7], Echo [10]. These bx tools are based on various approaches, such as graph transformations, bidirectionalization, update propagations [8] and can be used in different areas of application.

1.2 Problem Statement

Bidirectional transformation is an emerging concept. In the past, many efforts have been made by conducting international workshops, seminars and through experiments conducted by developers / bx community to identify its potential. Also, in addition to the development of bx tools and bx language, benchmarks are being created for bx tools for systematic comparison [4].

Although a significant amount of work has been done in this field, some basic problems still remain:

- Reachability to relevant communities is not significant due to the absence of a common vocabulary for bx across research disciplines [3]. Seminars are still conducted for exchanging ideas in different communities to define a common vocabulary of terms and properties for bx [2].

- Bx tools and their applicability is still not widely known even in the developers' communities. Due to the existing conceptual and practical challenges associated with configuring/trying a bx-tool for knowing its potential, using bx, bx-tools in building software systems, many developers and researchers are still using non-bx transformation tools to achieve properties which can be easily supported by bx-tools [3].
- Absence of a simple yet interactive bx tool demonstrator to depict the potential of *bidirectional transformation* over preferred non-bx tool demonstrators among developers' and researchers' communities [3].

1.3 Solution Strategy

To solve the problems as described in Section 1.2, in this thesis, my goals are as follows:

- Design and implement an interactive demonstrator.
- Spreading the basic concepts of bx to a wide audience and making them accessible and understandable.

An existing bx tool will be used as a part of the demonstrator to realize *bidirectional transformation*. The final prototype will be interactive and easily accessible to users to help them understand the potential, power and limitations of bx.

1.4 Structure

This document is structured as follows:

Chapter 1 (introduction) contains the introduction and motivation about the thesis with a solution strategy.

Chapter 2 discusses the related terminologies with respect to bidirectional transformation.

Chapter 3 describes the related work that has been done on bx in last few years and the related problems.

Chapter 4 explains the research work that I have done throughout my thesis along with research questions that I aimed at solving with my work.

Chapter 5 describes all the high-level concepts of my implementation work in brief with related diagrams.

Chapter 6 provides the in-depth details of each implementation layer along with UML diagrams.

Chapter 7 presents a walkthrough of the application from UI perspective.

Chapter 8 contains the feedback from user groups, evaluation results and learning goals(based on research questions).

Chapter 9 contains the feedback from user groups, evaluation results and learning goals(based on research questions).

Last chapter summarizes all the work which was done as part of this thesis and draws useful conclusions followed by future work.

2 Foundation

In this chapter, I am going to describe some commonly used terminologies with respect to bx so that the reader doesn't have any problem in understanding the further chapters.

As already mentioned, "bx is a technique to maintain consistency(synchronize) between two(or more) related models".

Model is an artefact or external representation of reality which reflects certain relevant aspects of a real system. It is denoted as "M".

- Artefact, because it is something artificial, not real.
- External representation, because it can be a drawing, a physical object or software generated on a computer.
- Certain relevant aspects, because it contains some properties related to the real system.

State denotes the condition of an artefact at any given point of time.

Transformation is the process in which an artefact undergo a change from one state to another. It is denoted as "X". Refer Figure 1.

Delta is the change done to one or more properties of an artefact which causes the transformation (X). It is denoted as δ.

State Space describes all the states of an artefact and all the deltas which lead from one state to another. Refer Figure 2.

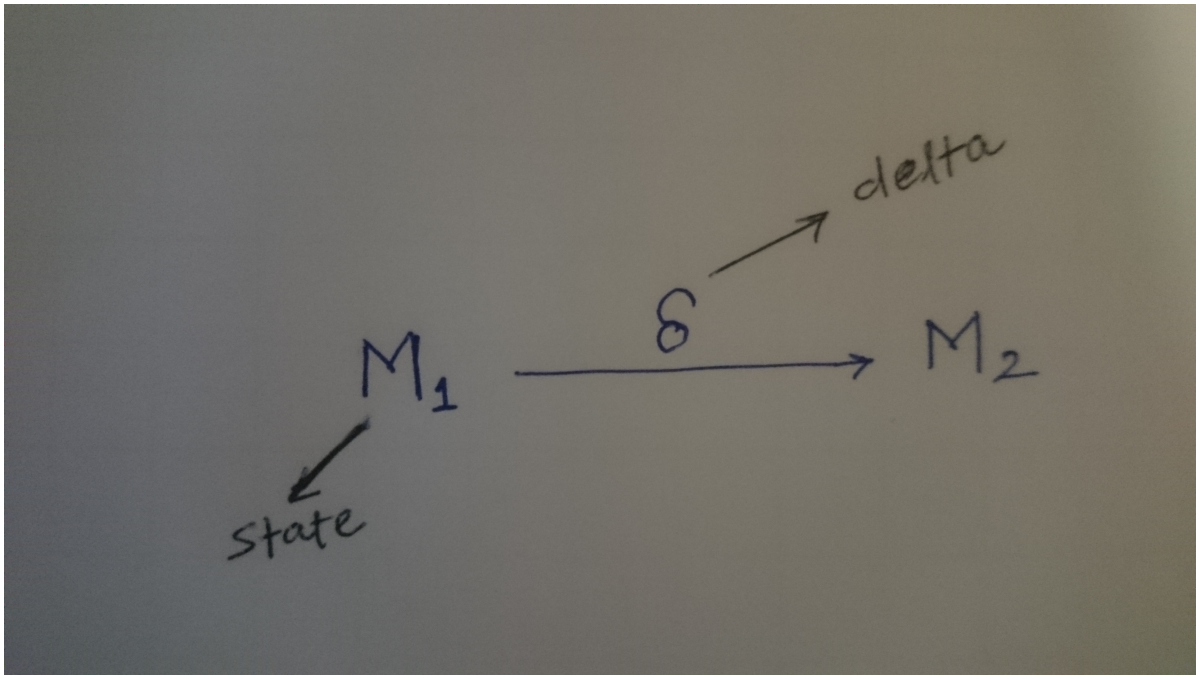

Figure 1: Transformation

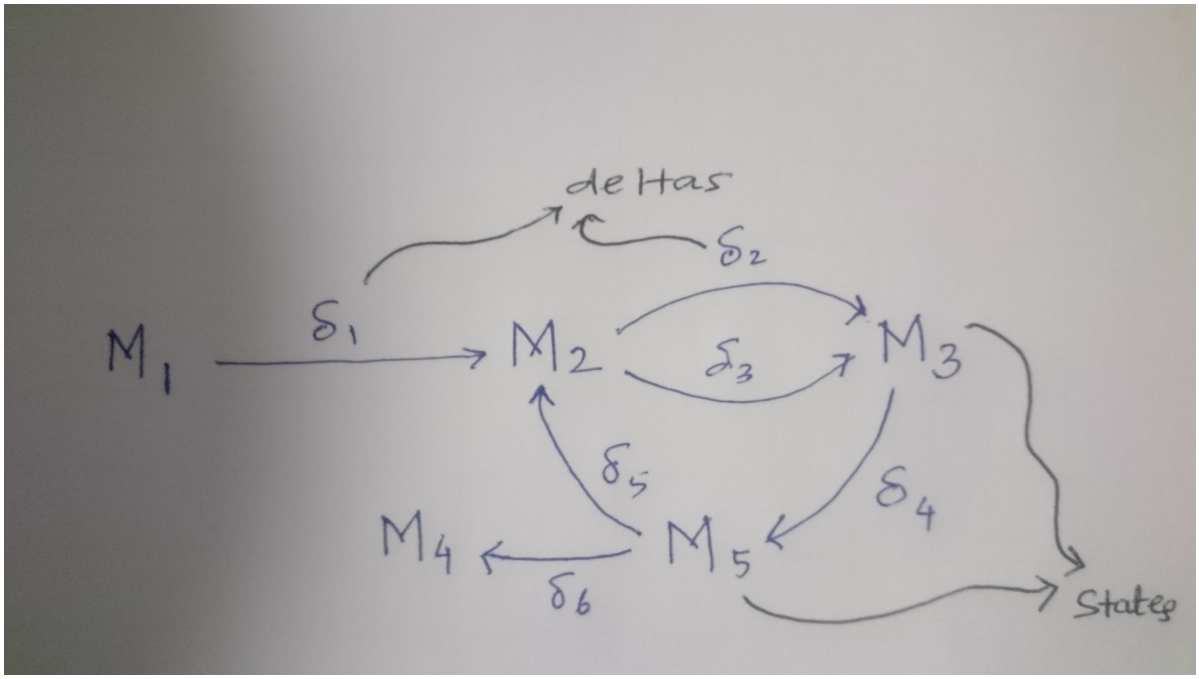

Figure 2: State Space

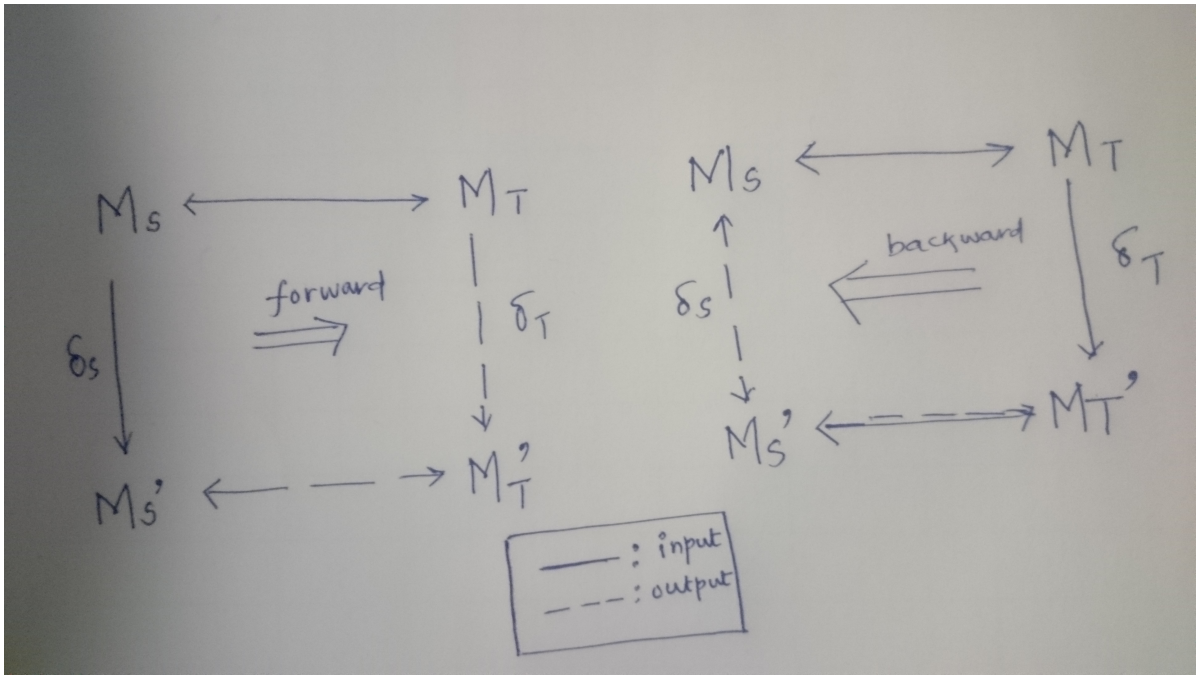

Figure 3: Bidirectional Transformation

Consistency In layman's term, consistent is a state which always involves 2 or more objects/artefacts but doesn't involve ambiguity between them. This is the most important part of the model transformation and the entire focus revolves around it. To be consistent with each other the models have to be inline with respect to their states. Changes in one model may or may not cause any change in other model but their states must not contain any contradiction. In model-transformation , the most important part is to

Unidirectional vs.Bidirectional Transformation Unidirectional is the simplest one out of all kinds of transformation. It always takes one type of input and produces same type of output [15]. The concept of consistency is very simple as the input model is consistent with the output model, only.
Whereas, bidirectional transformationis a pair of transformation which takes place in both forward and backward direction. One model can be input sometimes and output at some other time. The concept of consistency is relatively complex as more than one model and the models must be kept consistent. Source model (M_S) is transformed to target model (M_T) in forward transformation and vice-versa in backward transformation. It is denoted by "BX".
Figure 3 describes the concept in detail.

3 Related Work

This chapter sums up all the related work that has been done on bx. Section 3.1, 3.2, 3.3 and 3.4 describe the various ways that bx community and developer's group have tried to make the work done on bx visible to the world. Then, Section 3.5 explains the related problems.

Model transformation is a central part of Model-Driven Software Development [1] [2]. Bx community has been constantly doing research and development work in many fields to help people understand and increase awareness about bx. Nowadays, researchers from different areas are actively investigating the use of bx to solve a variety of problems. A lot of work has been done in terms of building usable tools and languages for bx. These tools can be used in various fields, for achieving *bidirectional transformation*. To understand these tools, several handbooks, tutorials and examples have been created so that users and developers can understand the core concepts.

3.1 Handbooks & Tutorials

As a part of our research, I have analyzed some tutorials and tools and below are my findings.

Anjorin et al.[7] present the concept for *bidirectional transformation* using Triple Graph Grammars (denoted by "TGG") [5]. To demonstrate their core idea and the usage of the tool, they have described an example by transforming one model (source) into another (target) through TGG transformations [5][6]. The whole tutorial is about 42 pages long which guides the user to get the example running through a series of steps. These steps include installing Eclipse[1], getting their tool as an Eclipse plugin, setting up the workspace, creating TGG schema and specifying its rule and much more. If the user is able to execute each step correctly, then finally he/she can view the final output. It took me 4 days to get the tool up and running.

We have analyzed a tutorial[12] on a bidirectional programming language BiGUL[11]. The core idea with BiGUL is to write only one putback transformation, from which the unique corresponding forward transformation is derived for free. The whole tutorial is about 45 pages long which includes a lot of complex formulas, algorithms, and guides the user to get the example running through a series of steps. These steps include installing BiGUL, setting up the environment, achieving bx through BiGUL's bidirectional programming and much more. If the user is able to execute each step correctly, then finally he/she can view the final output.

[1] *Integrated Development Environment (IDE) for programming Java*

3.2 Example Repositories

A rich set of bx examples repository [9] has been created based on many research papers. These examples cover a diverse set of areas such as business process management, software modeling, data structures, database, mathematics and much more.

A user can find relevant information about the examples on the respective web pages. Some of the examples are very well documented along with class diagrams, activity diagrams, object diagrams etc. and source code of a few examples are available as well.

3.3 Existing Demonstrators

Also, we have analyzed an existing demonstrator available along with the test cases of a domain-specific language, BiYacc [13] which is based on BiGUL [11].

Being an online demonstrator, a user can try it out instantly and check how it works. It doesn't require any installation or technical expertise to get the example running and also, it makes the features of bx noticeable.

3.4 Virtual Machines

Also, we have analyzed a web-based virtual machine, e.g., SHARE [14]. Basically, this is a web portal used for creating and sharing executable research papers and acts as a demonstrator to provide access to tools, softwares, operating systems, etc., which are otherwise a headache to install [14].

This provides the environment that the user requires to execute his/her tool or program. Hence, it reduces the overhead of a user for maintaining and organizing all software framework related stuff and simplify access for end-users.

3.5 Related Problems

Some associated problems that I found with the above paragraphs are as follows:

- Installation requires technical expertise and time consuming as the user typically has to setup and install the tool.
- Steps to get the example running needed technical expertise, e.g., In some cases, domain specific knowledge includes mathematics and specific coding language. What is

showcased and discussed is tied to a specific technological space and might not be easily transferable to other bx approaches.

- Not very helpful to understand what bx is before deciding which bx tool (and corresponding technological space) to use.
- The existing demonstrator's visual representation doesn't create interest in the target audience as it does not exploit the potential of using an interactive GUI and colors, etc. It just makes use of two text fields and is comparable to a console-based interface that is accessible online.
- The existing demonstrator is based on a rather technical example that might not be relevant, interesting, or convincing for a large group of potential bx users.
- For security reasons, virtual machines are not like other web portals where a user need to simply sign-up and can host/create/access data, rather it includes a series of request-grant cycle for getting access to an environment and hosting/managing data. Also, some actions require special authorization and take time to complete the whole process.

4 Requirements

This chapter explains all the research work that I have done before or during the actual implementation. Section **??** describes the phases I went through during my thesis work and the research involved. Afterwards, Section 4.2 deals with the actual research questions I am going to explore during my thesis. Then, Section 4.3 and 4.4 describes the learning goals for the user and what to expect from the thesis at the end respectively.

4.1 Research Phases

My work is based on an evolutionary case study[2] which focuses on designing and implementing a successful bx tool demonstrator. My entire work cycle is described in the following paragraphs.

Case Study Initially, the case study was based on the existing work/research done on bx, existing tools available in the market, flexibility in usage, time and technical expertise required to use these bx tools, and the implementation of these tools in different areas. With the initial study and knowledge gathered, I have formulated a few research questions as described in Section 4.2 in order to evaluate them in my thesis and improve the existing situation and the usefulness of a bx-tool based demonstrator.

Next step was to choose a bx-tool for my demonstrator. Based on the gathered information and taking account implementation related issues, I have finally chosen a bx-tool to be used as a part of the demonstrator to realize bx. Kindly refer Section 5.2 for detailed explanation on bx-tool selection.

Along with the research questions, I have also prepared some learning goals as described in Section 4.3 which the user needs to learn/understand about bx and the bx-tool while playing with the demonstrator.

Examples and Implementation Initially, I have constructed a few examples which can be implemented covering the requirements and showing the usability of bx tools through demonstrator. Then, taking account the availabilities of resources and usability factor, I finally chose the best suitable example to implement and build the final prototype. Section 5.1 describes list of all the examples.

Next step was to set up the entire application framework for implementing the demonstrator.

[2] *Case Study which keeps on evolving with time*

First, I did some research by going through materials on software design patterns and web application architecture. Then, I prepared a few proof of concepts(POC) for checking the feasibilty of the architecture designs before finalising my application framework. Kindly refer Section 5.3 for detailed explanation on finalizing the architecture design.

Evaluation To evaluate the demonstrator, I have conducted a few feedback sessions ....

Based on the feedbacks from the users, ... Kindly refer Section 8 for detailed explanation on evaluation and feedback.

Choices and Threats The implementation process and the final prototype was driven by many choices and threats. For example,

- selection of the bx tool and the most suitable example to implement has impact on deciding the usefullness of the demonstrator.
- seletion of the bx tool and the example to implement is more or less influenced by the ideas given by my supervisor.
- my decisions on designing the application's framework for implementation are impacted by the existing availability and usability issues of the finalized bx-tool.
- During evaluation, I might not have got proper feedback from my friends & colleagues due to my acquaintance and time constraints.

4.2 Objectives

The goal of this thesis is to explore the fundamental and technical challenges involved in implementing a demonstrator for bx tools.

This thesis aims at answering these main research questions:
RQ1 – What are the core requirements for implementing a successful bx demonstrator ?

RQ2 – What kind of interactivity and to what extent is it required in the bx demonstrator ?

RQ3 – Which goals can be particularly well addressed in a bx demonstrator and why ?

RQ4 – To what extent is such a bx demonstrator reusable?
RQ4 can be split into the following sub-questions:
RQ4.1 – Is the implementation of the demonstrator bx tool-specific ?

RQ4.2 – Is the implementation of the demonstrator example-specific?

RQ4.3 – What part(s) of the demonstrator can be reused in implementing a different example ?

All of my work is directly or indirectly related to the above research questions.

4.3 Learning Goals

Along with exploring the challenges as described in Section 4.2, I am also focussing on teaching some basic concepts to the user about bx in the process of trying/playing with the demonstrator.

What am I trying to teach?

- Bidirectional is not always bijective.
- Not all changes can be propagated. In this case, consistency needs to be preserved.
- In BX, Challenge is to avoid or minimise information loss.
- Current limitation - you can only change one side.
- Synchronisation is interactive (to handle non-determinism).

4.4 End Result

As my thesis is more focussed on the implementation of a demonstrator for consistency management based on a bx tool, following are the end results I am trying to achieve:

- Reduce the installation time of the bx tool by making the demonstrator available online. Hence, the user is just a click away to try the bx tool and needs nothing to install on his/her machine.
- Demonstartor should be interactive and fun to play with.
- User should be able to learn/understand the concepts of bx as described in Section 4.3.

5 High-Level Concepts

In this chapter, I am going to describe all high-level technical details realized during the implementation of the demonstrator. Section 5.1 describes all the examples that I have conceptualized before choosing the final one for the implementation. This is then followed by the description of the steps taken for selecting the bx-tool in Section 5.2. Afterwards, Section 5.3 deals with the decisions taken for finalizing the apllication's architecture design.

5.1 Choosing an Example

Due to the existing pain points with the bx tools, as described in Section **??** and to solve the problems as described in Section 3.5, the main idea is to design and implement an interactive bx tool demonstrator. I have constructed a few examples for implementation as follows:

Task Management This prototype can be used for allocating tasks in a team. It contains two views e.g., supervisor's view and employee's view. A Supervisor can allocate tasks to their subordinates. An employee can view the tasks assigned to him. Then the task will go through a life cycle as the work progresses, i.e., Assigned, In Progress, Testing, Done. Supervisor's view shows aggregate information from multiple projects and multiple employees, but does not contain detailed information, e.g., tasks have fewer states than for assigned employees. Bx rules control how updates are handled and states are reflected in the different views of the project, e.g., the employee's view will be updated for each state change, whereas the supervisor's view is only updated when a task is completed and not for intermediate changes.

Quiz This prototype can be used for an online quiz game. It contains two views e.g., administrator's view and participant's view. There will be a large set of questions related to different areas, e.g, history, geography, politics, sports, etc. The administrator can select the areas from which the questions will be shown to the participant and initiate the game. The participant can override the selection of the areas and start the quiz. Randomly questions will be shown to the participant from the selected areas with 4 options. The administrator's view contains less information than the participant's view, e.g., only the result of each question will be shown to the administrator, whereas participant can see questions along with its options. As soon as the participant chooses the answer to any question, bx rules control how updates are handled and states are reflected in the different views of the project.

Playing with Shapes It contains two views e.g., low-level view (depicts UI[3] for low-level language, i.e., UI with less functionality) and high-level view (depicts UI for high-level

[3] *short for User Interface*

language, i.e., UI with more functionality). User will draw a geometric shape, i.e., triangle / square / rectangle / circle with some notations similar to the shape on the low-level view and if the notations are correct, the high-level view tries to recognize the shape and draws it with default parameters and vice-versa. Basically the transformation will happen between a low-level language and a high-level language and bx rules control how updates are handled and states are reflected in the different views of the project. In high-level view, more functionalities will be present, i.e., moving one shape from one place to another, creating a clone of an existing shape, etc. which is not possible in low-level view.

Arranging a Kitchen It contains two views e.g., low-level view (depicts a grid structure containing blocks) and high-level view (empty space which depicts UI for kitchen). High-level view has more functionalities such as creating/ deleting/ moving an kitchen item, etc. out of which only a few will be available in low-level view. User will create/ delete/ move a kitchen item, i.e., sink / table on the high-level view and if changes done on the high-level view are according to the rules defined in the bx tool then items will be reflected on the low-level view with same colored blocks and vice-versa. Basically the transformation will happen between a low-level language and a high-level language and bx rules control how updates are handled and states are reflected in the different views of the project.

5.2 BX Tool Selection

5.3 Concrete Design Decision and High-Level Architecture Information

High-Level Architecture diagram of my prototype is given in Figure 4.

5.3.1 View

It contains the entire UI elements and its logic.

5.3.2 Controller

5.3.3 Model

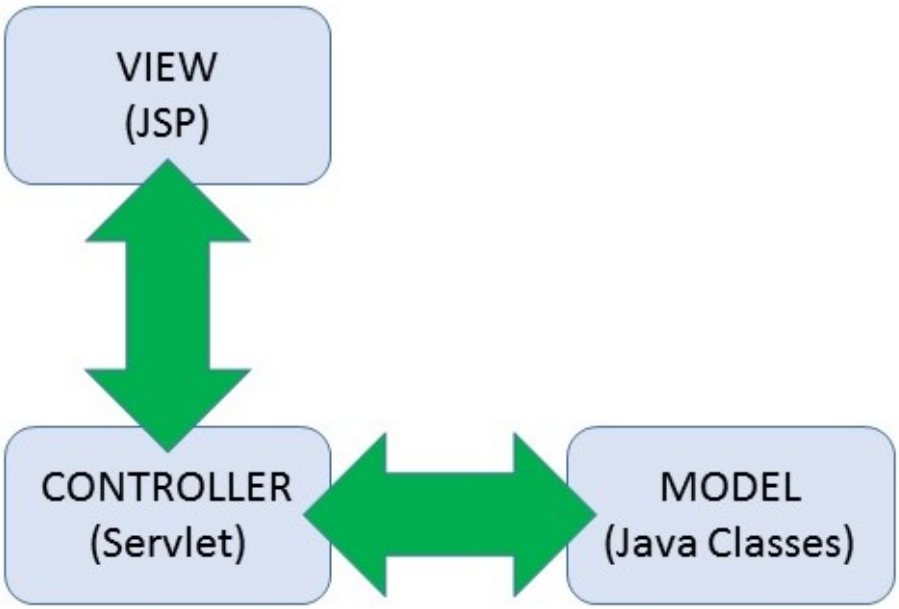

Figure 4: High Level Architecture Diagram

6 Low-Level Concepts

6.1 UML Diagrams

6.1.1 Component Diagram

Component diagram of my prototype is given in Figure 5.

I am using Model-View-Controller (MVC) pattern for my application framework. On the top, `View` (Web Browser) component is present which contains a graphical user interface and functionalities that belong to the user. With the changes on `View` component, data are being provided through interface `IDoAnalysis` to the `Controller` component and after the calculations are done, the results are sent back to the `View` through interface `IProvideResults`. Both `View` and `Controller` resides on the same machine. `Model` (Bx Tool) component encapsulates and manages the state of all models by communication with the `Controller` through interface `IRules`.

6.1.2 Class Diagram

6.1.3 Sequence Diagram

6.2 Core Details

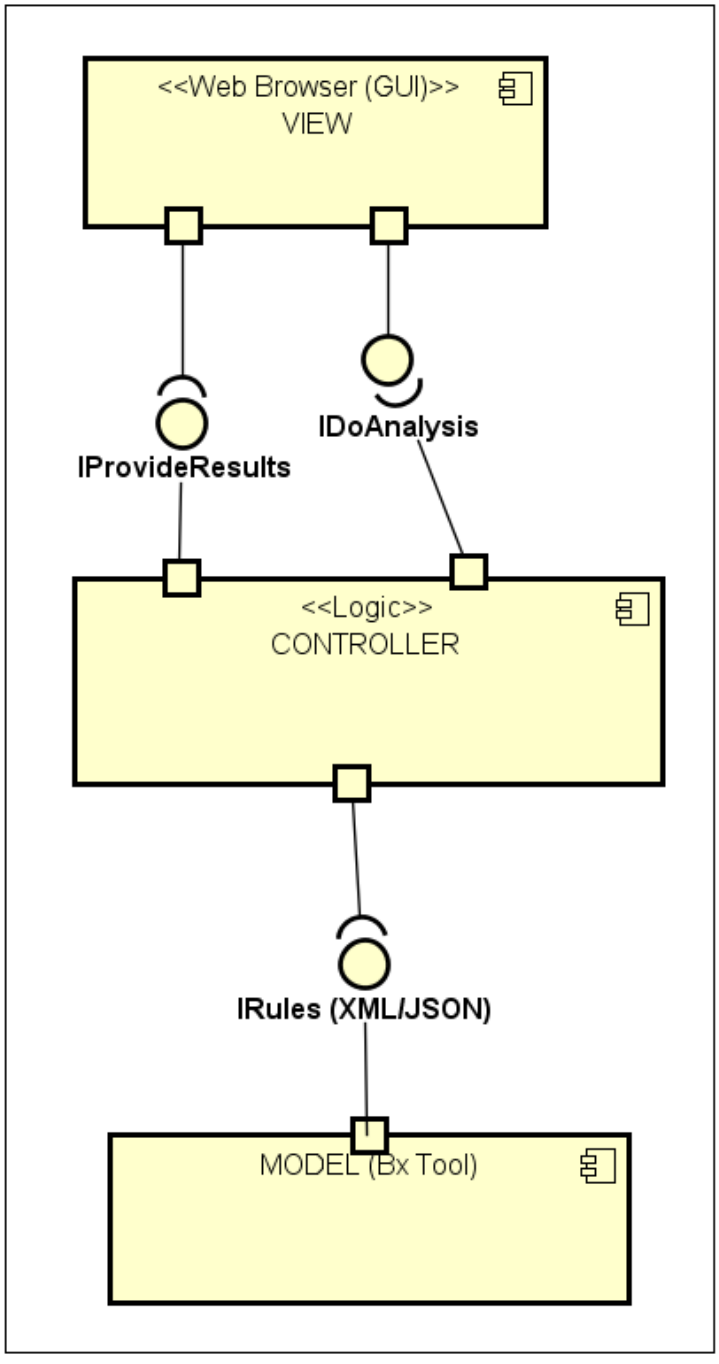

Figure 5: Component Diagram

7 Application

7.1 Application Walkthrough

8 Evaluation

8.1 Discussion and Feedback

8.2 Evaluation and Results

9 Summary and Way Forward

9.1 Conclusion

9.2 Future Work

10 Appendix

The Appendix can be used to provide additional information, e.g. tables, figures, etc.

List of Figures

List of Tables

References

[1] K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, A. Schürr, and J. F. Terwilliger, *Bidirectional Transformations: A Cross-Discipline Perspective*, GRACE International Meeting , Shonan, Japan, 2008. 1, 6

[2] Z. Hu, A. Schürr, P. Stevens, and J. Terwilliger, *Dagstuhl Seminar* #11031 on Bidirectional Transformations "bx", Dagstuhl Reports, Vol. 1, Issue 1, pages 42-67, January 16-21 , 2011. 1, 6

[3] J. Gibbons, R. F. Paige, A. Schürr, J. F. Terwilliger and J. Weber, *Bi-directional transformations (bx) - Theory and Applications Across Disciplines*, [Online]. Available: https://www.birs.ca/workshops/2013/13w5115/report13w5115.pdf. 1, 2

[4] R. Oppermann and P. Robrecht, *Benchmarks for Bidirectional Transformations.* Seminar Maintaining Consistency in Model-Driven Engineering: Challenges and Techniques, University of Paderborn: Summer Term 2016. 1

[5] A. Schürr. *Specification of graph translators with triple graph grammars.* In E. W. Mayr, G. Schmidt, and G. Tinhofer, editors, *Graph-Theoretic Concepts in Computer Science, 20th International Workshop, WG 94*, volume 903 of LNCS, pages 151-163, Herrsching, Germany, June 1994. 6

[6] A. Bucaioni and R. Eramo, *Understanding bidirectional transformations with TGGs and JTL*, in Proc. of the Second Workshop on Bidirectional Transformations (BX 2013), no. 57, 2013. 6

[7] A. Anjorin, E. Burdon, F. Deckwerth, R. Kluge, L. Kliegel, M. Lauder, E. Leblebici, D.Tögel, D. Marx, L. Patzina, S. Patzina, A. Schleich, S. E. Zander, J. Reinländer, and M. Wieber, *An Introduction to Metamodelling and Graph Transformations with eMoflon. Part IV: Triple Graph Grammars.* [Online]. Available: https://emoflon.github.io/eclipse-plugin/release/handbook/part4.pdf 1, 6

[8] BX Community, [Online]. Available: http://bx-community.wikidot.com/ 1

[9] BX Community, [Online]. Available: http://bx-community.wikidot.com/examples:home 7

[10] N. Macedo, T. Guimaräes and A. Cunha, *Model repair and transformation with Echo*. In Proc. ASE 2013, ACM Press, 2013. 1

[11] Hsiang-Shang Ko, Tao Zan, and Zhenjiang Hu, *BiGUL: A formally verified core language for putback-based bidirectional programming.* In Partial Evaluation and Program Manipulation, PEPM'16, pages 61-72. ACM, 2016. 6, 7

[12] Zhenjiang Hu and Hsiang-Shang Ko, *Principle and Practice of Bidirectional Programming in BiGUL*, Tutorial [Online]. Available: http://www.prg.nii.ac.jp/project/bigul/tutorial.pdf 6

[13] *BiYacc, tool designed to ease the work of writing parsers and printers*, [Online]. Available: http://biyacc.yozora.moe/ 7

[14] *SHARE - Sharing Hosted Autonomous Research Environments*, [Online]. Available: http://is.ieis.tue.nl/staff/pvgorp/share/ 7

[15] *Wikipedia. "Model transformation."* Retrieved March, 2017, from https://en.wikipedia.org/wiki/Model_transformation. 5