Analysis of Bret Victor's principle and its application for Empirical Modelling

1264707

Abstract

Bret Victor, who gave a presentation at CUSEC 2012, described his brilliant principle of programming interface. This shares a lot of similarities with current Empirical Modelling processes and he described a lot of features that would be very useful for Empirical Modelling to have. Motivated by this fact, this paper analyses his presentation, and suggest ways of implementing these features to extend current Empirical Modelling tools in order to accommodate more interesting ideas.

1 Introduction

Bret Victor, an ex-Apple employee who specializes in designing interfaces, gave an inspirational speech at CUSEC (The Canadian University Software Engineering Conference) 2012 about his principle of design in January 2012[1]. His view on designing stage shares a lot of similarities with the modelling stage of Empirical Modelling. In particular, one of the most important similarities that they share is the fact that changes made are reflected immediately on the model (or the design, in Bret Victor's case). In this presentation, he described a lot of techniques he implemented on the coding interface he wrote. They are not only relevant, but also very useful to accommodate this idea of immediately seeing changes in the design. This brings up a new idea, which is to analyse his techniques and attempt to apply them to Empirical Modelling.

Naturally there are differences between the programming process described in Bret Victor's presentation and Empirical Modelling. The differences will result in a lot of the features that perhaps cannot be done with current Empirical Modelling tools, or may require a lot of efforts in redesigning the tools, or simply are not relevant.

Both the differences and similarities will be analysed to give an insight of how Bret Victor's ideas may fit into Empirical Modelling. This will in turn help providing insight of a possible way to improve current Empirical Modelling tools.

2 Similarities and Differences

Firstly, it is worth comparing what Bret Victor described in his presentation with the current Empirical Modelling tools, and also the philosophy of Empirical Modelling. This will give a clearer picture of the direction we will be heading when implementing the features Bret Victor described.

2.1 Differences

In the first part of Bret Victor's presentation, he described the application of his techniques on a piece of JavaScript codes. He used his tools to go back and forth and change different parameters on the code and saw immediate changes of the design (Victor, 2012). This is different from the way we change observables in current Empirical Modelling tools. In Empirical Modelling tools like TKEDEN or JSEDEN, an observable is changed using the interpreter window. A line of script is added into the window that redefines the observable giving it a new value. This is why Empirical Modelling is said to be using definitive scripts. This is one of the problems we have to keep in mind, as we will see later, when we want to implement some features that Bret Victor described.

Another thing to keep in mind is the fact that he described his techniques applied on a piece of procedural programming code (using JavaScript). This is different from Empirical Modelling, which relies on ODA (Observable, Dependency and Agent) to construct models[2]. It also means that there is unpredictability in models behaviours instead of being predictable like in the design. This is largely due to

[1] The full presentation can be accessed online on his website http://worrydream.com/

[2] According to Warwick DCS website which can be accessed online on: http://www2.warwick.ac.uk/fac/sci/dcs/research/em/intro/principles/

the dependency and agent components in Empirical Modelling: They act differently with each different setting of the model. This affects the way we adjust the implementation.

2.2 Similarities

First of all, programmers can already see immediate changes in Empirical Modelling models after inputting a new script to the interpreter window. This is not something usual in procedural programming, and this is something that Bret Victor actually seeks for procedural programming design (Victor, 2012). However, as he successfully devised a method to achieve that, he also elaborated his techniques to include many different exciting features. This is the main motivation for analysing his work, to try to implement new exciting features to our current Empirical Modelling tools.

Secondly, his approach to design, which he described as "nothing is hidden from the designer" is similar to that of Empirical Modelling models making process (Victor, 2012). In Empirical Modelling, programmers have to be able to see the result of each input script, in order to build up a full model. It is reassuring that his ideas and his work are very relevant to what we have in Empirical Modelling.

3 Features

In Bret Victor's presentation, he described many different features implemented to accommodate his idea. We can break down the features one by one to find suitable features for Empirical Modelling.

3.1 Immediate Change

The highlight of Bret Victor's presentation is his talk about immediate changes of the designs reflected whenever a parameter is changed. According to him, this is very important in the creative process, as the creators need to see the changes they made immediately instead of keeping track in their heads (Victor, 2012). This supports his principle of having "nothing hidden" in the creative process. Using current Empirical Modelling tools such as JSEDEN or TKEDEN, we can already achieve this. Whenever a script is added, whether to redefine a dependency, modify an observable or to create new ones, the model changes accordingly and immediately.

Bret Victor enhanced this feature further by adding sliders to his design interface. The sliders allow creators to change the parameters very quickly. The main advantage of this method is it speeds up the creative process tremendously. Instead of having to guess what the value of the parameter, creators can move the slider up or down to continuously see the changes, then decide on what they want their designs to look like. Another advantage of this enhancement is that it sometimes may have a secondary effect. As demonstrated in his presentation, Bret Victor pointed out that creators might come across more ideas when using such convenient tools in their creative process, like how he came across the idea of animation when using the slider to change the value on his demo design. While this may not happen every time, this is certainly a useful aspect of this enhancement.

In Bret Victor's tools, creators could modify the codes directly from the source. However in Empirical Modelling Tools, there is no "source" per se, but instead the scripts are added dynamically via the interpreter window. Even if the script is meant to load out a file, i.e. a "source", the file is actually a lot of saved scripts. One of the solutions is to redesign the current tools to incorporate the display of each parameter (Observable, Dependency, Agency) and then make the slider work from there. In JSEDEN, the parameters are already displayed so the tools can be extended from there. Another solution, which is less invasive, may be creating an Empirical Modelling tool to solve this problem. This tool consists of multiple sliders and users can assign each slider with each parameter by defining a dependency between them. One problem with this approach is that after the design of the model is finished, the sliders will still be there unless there is some other methods to remove them. It can be an advantage, as it can be thought of as another approach to Empirical Modelling, but it can also be a disadvantage as it takes up space on the display canvas. Either way it is a much simpler way of implementing this feature than redesigning the tools, and it will still work across different tools as well as time change, as it is an EM tool not as part of any particular environment tool.

It should be noted that this feature works mostly on redefining Observables. For Dependencies and Agencies, they may not be numerical values. However, we can be creative about how to apply this immediate changes, for example we may have autocompletion for different values for users to scroll through them instead. It may be difficult to implement but can be useful and can be considered in the future when the development of EM tools are more complete.

3.2 Code Mapping

Another important feature that Bret Victor implemented in his interface is the ability to map code and design directly. Creators can find out which line of code draw the design or what the code does by

hovering on either the design or the line of code. This helps the creators navigate through their designs easily. Not only that, if a design is transferred to another person, then this person may be able to navigate through the design easily without having to take much time to learn about it. According to Bret Victor, this is a further solution to the creators not having to keep things in their heads.

This feature is certainly very useful for Empirical Modelling tools to have. If we look at JSEDEN, then we may be able to exploit browser's developer tools such as Google Chrome Developer Tools to map lines of JavaScript code to the model. However, there are a few problems with this. One is that the creators will probably have to be adept at JavaScript to understand how to modify it. It can be argued that because of JSEDEN hybrid approach nature, the creators should have some knowledge about JavaScript to make full use of the tool. While this may be true, there might be some problems with understanding the JavaScript due to most of the design is usually made up using EDEN scripts, and some symbols may be translated differently. We also have a problem with translating between EDEN scripts and JavaScript to make full use of this feature. Therefore it is useful to create a tool to match the EDEN scripts with the model instead.

However, this brings up another problem. With current JSEDEN, the only place where EDEN scripts are displayed is the History Window. It displays every inputs of the current model, which means it contains redundancies, i.e. the old definitions that may be overwritten by new scripts. So if we want to somehow display the scripts to match with the model, we have to somehow get rid of the redundancies. On the other hand, with the current JSEDEN implementation, we can view the current definitions from the ODA frame. It may be more viable to implement the mapping from this ODA frame to the model instead, because what is really significant in this feature is the ability to identify the functionalities of the definitions. Therefore we can just map the definitions to the model instead of trying to extract the scripts.

There is currently no known tool for Empirical Modelling to map the model to its definitions. This feature is definitely useful as in larger models; it is often difficult to match the model with its own definitions. There may be many observables with similar names, or the names may not be that obvious for whoever is looking at the model. Having this feature will save a lot of time in the creative process especially for larger models.

Implementation of mapping features is not an easy task. It may require recoding the whole Empirical Modelling tool to accommodate such feature. It is a very good idea for future projects.

3.3 Other features

There are other features mentioned in the presentation. Some of which may be useful for our tools, but some might not be. To be thorough with the analysis, they will be discussed here.

One of the major features that Bret Victor described in his presentation was the ability to manage time. In his presentation, he demonstrated a tool with an example of a platform game design. The tool helps the creators tell the "future" of the game state by recording a number of inputs, then pause the game and rewind. His tool then trails the changes across the screen and creators can modify parameters and the trail will change accordingly. This feature is especially useful when designing something that is constantly changing with time and cannot be easily seen.

There are a few problems with incorporating this feature into Empirical Modelling. The main goal of the design process is to create behaviours of the objects that the creators wanted. If this were also the goal of Empirical Modelling, then this would be a great addition to the current features. However, in Empirical Modelling, the goal is not to design behaviours, but to understand the behaviours of the models with given criteria (i.e. given Observables, Dependencies and Agents). Therefore it may not always be appropriate for such tools to be in Empirical Modelling. Another problem with Empirical Modelling is that a lot of the models also have different input options, to accommodate the goal of understanding the situation that it modelled after as well. Therefore it may not be appropriate to design behaviours with constant inputs like we see in his demonstration. On the other hand, it may be a useful tool to use here and there, but with the complexity of the tool, versus its low relevance, it may not be worth the effort to implement such thing.

Another small feature that Bret Victor mentioned was the ability to auto-complete the definitions. In his presentation, he demonstrated the ability to auto-complete a line of code, then scroll through it to see what it does. Although this is not a major feature, it may be a nice addition to our current tools. As mentioned above, it can be applied to other aspects such as auto-completion of dependencies as well. However, it may be complex and with the current development stage of our Empirical Modelling tools, it is best to devote our efforts to other major features first.

4 Conclusion

After analysing Bret Victor's work, we can see many features that can be applied to our current Empirical Modelling tools. This paper has brought up some insight of such features and ideas of possible ways to implement them. This may be an important step of extending Empirical Modelling tools, and it may change the directions that Empirical Modelling tools are heading.

5 Future Work

In the future, the feasibility and small details should be considered. The implementation can be carried out in JS-EDEN, as its hybrid approach results in a lot of flexibilities. However, it is appropriate that in the future if there are other EM tools developed and written in other languages, these features should be kept in mind to enrich their interface.

Acknowledgements

Acknowledgement should go to Bret Victor himself, for providing us with such an inspirational presentation that can be seen on video on his website. I would also like to thank professor Meurig Beynon for providing me with directions for this paper.

References

Victor, B. 2012. The Canadian University Software Engineering Conference. *Inventing on Principles*. January 20. [keynote]