SimNav: Simulink Navigation of Model Clone Classes

Eric J. Rapos, Andrew Stevenson, Manar H. Alalfi, James R. Cordy
School of Computing, Queen's University
Kingston, Ontario, Canada
{eric, andrews, alalfi, cordy}@cs.queensu.ca

***Abstract*—SimNav is a graphical user interface designed for displaying and navigating clone classes of Simulink models detected by the model clone detector Simone. As an embedded Simulink interface tool, *SimNav* allows model developers to explore detected clones directly in their own model development environment rather than a separate research tool interface. *SimNav* allows users to open selected models for side-by-side comparison, in order to visually explore clone classes and view the differences in the clone instances, as well as to explore the context in which the clones exist. This tool paper describes the motivation, implementation, and use cases for *SimNav*.**

I. Introduction

Clone detection in Simulink models can serve a number of purposes, such as finding model patterns and anti-patterns, examining evolution between versions, enforcing consistency and standards, and many other applications in model maintenance and quality assurance. Simone [2] is a text-based near-miss (Type 3) clone detector for Simulink model clones based on the NiCad [9] code clone detector. The native NiCad output of Simone, consisting of text-based HTML and XML reports, can be difficult for Simulink model developers to understand and relate to the visual Simulink graphical models they normally work with. *SimNav* provides a bridge between these two representations that is intended to bring the clone detection results of Simone directly into the developers' own Simulink modelling environment where they can explore, investigate and act upon clone detection results in the context of their own familiar workspace.

Emergent patterns in Simulink models found through clone detection [4], whether they are exact clones or near-miss clones, provide developers with useful information about a collection of models. Similar model sections can be indicative of the reuse of model sections, either intentionally or unintentionally, which can provide developers with the opportunity to create new library blocks, which can replace the clone instances. Model clones can also uncover instances of documented design patterns or anti-patterns [11], assisting in overall quality improvement.

Simone is also useful for studying evolution of Simulink models and model clones [10]. By using Simone on multiple versions of the same model, it is possible to identify sections of models that remain unchanged from version to version, or change minimally (near-miss results), as well as observing how clone classes evolve over versions. This information is useful as it reveals areas of models that remain consistent over time, and the inverse set can be used to find sections of the model that are prone to changes in each version, which is also useful for developers as it indicates sections that may be prone to the introduction of bugs in subsequent versions.

Each of these uses reveals information that can be used by developers to improve the model maintenance process. The detection of patterns can lead to the creation of library blocks for commonly occurring exact clones, and the detection of near-miss clones can assist in Simulink variability modelling [3]. The detection of clones over model versions reveals information that will help in the maintenance over time by identifying sections of models that require additional attention and testing, as they evolve more frequently than others, as well as sections that evolve less frequently, or not at all. *SimNav* assists in bringing the results of clone analysis directly into the environment where these tasks are carried out and actions can be taken to address issues directly.

The next section of this paper examines related approaches to presenting clone detection results. The remainder of the paper presents the implementation of *SimNav*, detailing the technical aspects, as well as presenting several use cases, accompanied by screenshots of the tool.

II. Related Work

CloneDetective [6] is a tool capable of performing clone detection on Simulink models and presenting the results to users in a graphical interface. CloneDetective reproduces the models and displays them in a custom stand-alone tool, rather than directly within Simulink, which requires that developers learn a new interface and precludes the possibility of working with the models directly.

ModelCD [8] presents another model clone detection tool for Simulink models. ModelCD uses an algorithm called *aScan* to find approximate clones, which are analogous to Simone's near-miss clones, and is also capable of finding exact clones using another algorithm called *eScan*. ModelCD allows for the viewing of clones within Simulink using a Matlab script that opens and colours the clones in Matlab, but the actual clone detection results are not available in Simulink and the viewed models are copies of the originals. The functionality of ModelCD is somewhat similar to opening individual models using *SimNav*, but without the ability to explore clone classes, similarity, and instances directly in the Simulink workspace.

The Naive Clone Detector [7] is an exact clone detector for Simulink models. Similarly to the other approaches discussed, the results are not presented directly in Simulink, but rather

a Matlab Connector is provided. The Naive Clone Detector is capable of presenting the clone detection results, however the Matlab Connector must be used to separately open the models in Simulink.

Model Quality Assessor [5] is another tool used to view clone detection results (referred to by the authors as clone inspection). Model Quality Assessor is an external tool capable of visually displaying the clones that were detected, but lacks the direct integration with the Simulink environment provided by *SimNav*.

III. Implementation

One of the main goals of our implementation was the integration with the Simulink environment; our tool needed to be familiar to Simulink model developers without training. As such, all development was done directly in Matlab, making use of the various libraries and functionality provided.

The remainder of this section details specific design choices in the areas of graphical user interface design, filtering and focussing methods, and outlines our process for industrial feedback.

A. Graphical User Interface Design

The SimNav graphical user interface (GUI) itself is simply designed to avoid cluttering and to ensure that users can easily identify how to perform desired tasks. The main portion of the GUI is a table which displays the clone classes that were detected by *Simone*. The table has four columns: (i) Clone Class, (ii) Similarity, (iii) Subsystem, and (iv) Model File. The Clone Class number is a unique ID assigned to each clone class sequentially, and used to reference a clone class. The Similarity value indicates the similarity for a particular clone class as a percentage. The Subsystem column identifies the full path to the subsystem where the clone instance occurs. And finally, the Model File column indicates which model file the subsystem is located in.

Beyond the table with its four columns, there are six controls along the top of the GUI: (i) Load Report, (ii) Open Selected, (iii) Close All, (iv) Deselect All, (v) Clone Type, and (vi) Similarity. *Load Report* will open a dialog box, which is used to select a *Simone* report to load into *SimNav* for further inspection. *Open Selected* will open all selected clone instances (instances are selected using a standard Ctrl+Click method, or by selecting an entire clone class by clicking on its ID) and tile them on the screen. *Close All* will close all opened models, and *Deselect All* will remove highlighting of any selected models. Alternatively, clicking on another clone class ID will close all open models, deselect all selected models, and select the new clone class. The *Clone Type* dropdown is used to filter the displayed results by the clone class type, which will be explained further in Section III-B. The Similarity control is used to filter the clone classes by their similarity; this is done by selecting a lower and upper bound on the similarity using the arrows on the selector boxes, and the filtering takes place after a 1 second delay, allowing the user to adjust both values without performing the sort multiple time. This filtering process will also be explained in Section III-B. Figure 1 shows all of these columns and controls in the *SimNav* toolbar.

Matlab's Graphical User Interface Design Environment (GUIDE) [1] provided us with the perfect platform for creating a GUI that would work within the Matlab/Simulink setting, while still allowing us the flexibility to implement our own functionality and features. GUIDE allowed us to create our GUI using a simple drag and drop interface where we were able to add buttons and other controls, change their attributes, and assign function calls to any events where a response from the tool was necessary. The final GUI for *SimNav* is simple, easy to navigate, and provides the desired functionality.

B. Filtering

SimNav allows for two types of filtering of the presented clone classes: (i) filtering based on the clone class type, and (ii) filtering based on the similarity of the clone class. This section will describe the implementation of these two types of filtering.

The first type of filtering is based on the concept of a clone type. As part of the refinement process, it became evident that the clone analysis would be useful to find clones within a single model, as well as across model files. As part of this, we developed the concepts of *internal clone classes* and *external clone classes*.

Internal clone classes are clone classes that contain more than one model instance from within the same model. Any clone class that has two or more instances from the same model file would be displayed during an *Internal Only* filter. An *internal clone class* may contain model instances from more than one model file, but each model file must have at least two instances, and any singleton model files are removed. As part of this filter, the model instances are sorted by model file name, in order to see how many instances occur in each file. Figure 2 shows an example of an *internal only* filter (center) performed on the full set of clone classes (left). In this example Clone Class 2 is filtered out as it contains only one instance from each of two different models, and the singleton model from Clone Class 3 is removed, leaving only the subsytems where there exists more than one instance from the model file.

External clone classes are clone classes that contain only model instances from different models. Any clone class that has at most one instance from a model file would be displayed during the *External Only* filter. Again, Figure 2 shows an example of this, where performing this filter (right) removes Clone Class 1 from the full set of clone classes (left) as it contains two instances from the same model file.

The default sort for *SimNav* is the *Internal & External* sort, which shows both types, and all instances in each clone class.

The filtering based on clone class is done at the time of loading the report. When the report is loaded into *SimNav*, three tables are created - one for each of the types of sorts. The tables are constructed by iterating through the input file, and constructing the 4-column table that is displayed within the GUI of SimNav. Each clone class is selected and all of

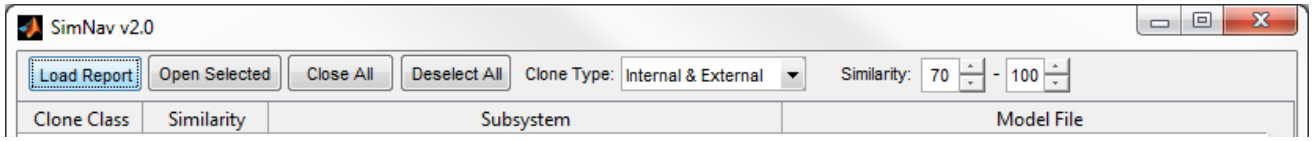

Fig. 1. The *SimNav* Toolbar

Clone Class ID	Similarity	Subsystem	Model File
1	85	ModelA/System1	ModelA.mdl
		ModelA/System2	ModelA.mdl
2	90	ModelA/System3	ModelA.mdl
		ModelB/System1	ModelB.mdl
3	80	ModelA/System4	ModelA.mdl
		ModelA/System5	ModelA.mdl
		ModelB/System2	ModelB.mdl
		ModelC/System1	ModelC.mdl
		ModelC/System2	ModelC.mdl
		ModelD/System1	ModelD.mdl

Internal & External Clone Classes

Clone Class ID	Similarity	Subsystem	Model File
1	85	ModelA/System1	ModelA.mdl
		ModelA/System2	ModelA.mdl
3	80	ModelA/System4	ModelA.mdl
		ModelA/System5	ModelA.mdl
		ModelC/System1	ModelC.mdl
		ModelC/System2	ModelC.mdl

Internal Only

Clone Class ID	Similarity	Subsystem	Model File
2	90	ModelA/System3	ModelA.mdl
		ModelB/System1	ModelB.mdl
3	80	ModelA/System4	ModelA.mdl
		ModelA/System5	ModelA.mdl
		ModelB/System2	ModelB.mdl
		ModelC/System1	ModelC.mdl
		ModelC/System2	ModelC.mdl
		ModelD/System1	ModelD.mdl

External Only

Fig. 2. Clone Type Examples: Internal & External (left), Internal Only (center), and External Only (right)

the information is added to the table representing the default sort of *Internal & External*, then the clone class is explored to see which of the other two types of filtering it matches, and is added to one of those tables as well. Once the three tables are stored in memory, the default filtering is displayed on the GUI, and the other two are readily available for quick replacement, meaning that no matter how many times the view is changed, the filtering is only conducted once when the report is loaded.

The second type of filtering that *SimNav* is capable of is filtering based on the similarity of the clone class. The default filtering is from whatever the lower threshold of the particular set of models is, which is identified at load time, to 100%. However it may be useful to filter out uninteresting results, such as those that are too similar or too dissimilar. Without wanting to impose too many restrictions on the filtering, we wanted it to be as free as possible, meaning that the user is able to choose a similarity range down to a granularity of a single percent.

Unlike the clone class type filtering, the similarity filtering is not done at the load time, and is done on the fly as the user selects the filtering they would like to be displayed. Once a user has selected a similarity range using the selector boxes, the upper and lower bound are passed to the filtering function to perform the filter, along with the current clone class filtering type. The filtering function will create a fresh copy of the full table for the currently selected clone class type (created at load time) and will iterate through each clone class, removing those that are outside of the set bounds. Once each clone class is either included or excluded, the resulting set of clone classes is displayed in the GUI.

C. Industry Feedback

Throughout the development process, we collaborated with our industry partners, presenting iterations of the tool, and requesting feedback on both functionality and GUI design. This section will detail some of the feedback and how it was incorporated into our design.

The first major piece of feedback that came from our industry partners was a request to be able to see only the clone classes that contain clones within a single model file. This was a use case that we had not previously considered, which led to the introduction of the filtering based on clone class types.

Similarly, the other type of filtering, based on similarity, was also a result of feedback from our industry partners. Initially, we would only present all of the results, but it was communicated to us that it would be useful to exclude some of the results, some of the time, meaning that it would be good to allow the user to choose for themselves which ranges of similarity they wanted to display.

Beyond the two filters, our industry partners also provided us with minor feedback relating to GUI design; a few requests that would make the GUI more intuitive and responsive to the way workflow would normally occur.

The collaboration with industry has provided us with additional insight into the use cases for *SimNav* which allowed us to further develop the tool into something desired for use in industry, and the positive feedback about the Simulink integration validated our initial goals.

IV. Use Cases

The following use cases show basic functionality of *SimNav*, using a combined set of open-source automotive models, which we first ran Simone on to obtain clone detection results. The models come from a number of open-source projects, and can be found on the *SimNav* page along with the download of *SimNav*.

A. Viewing Clone Class Reports

The first use case for *SimNav* is the viewing of the clone class reports generated by Simone. As part of the clone detection process, Simone produces a CSV file, which contains the clone classes, and the instance models. *SimNav* provides

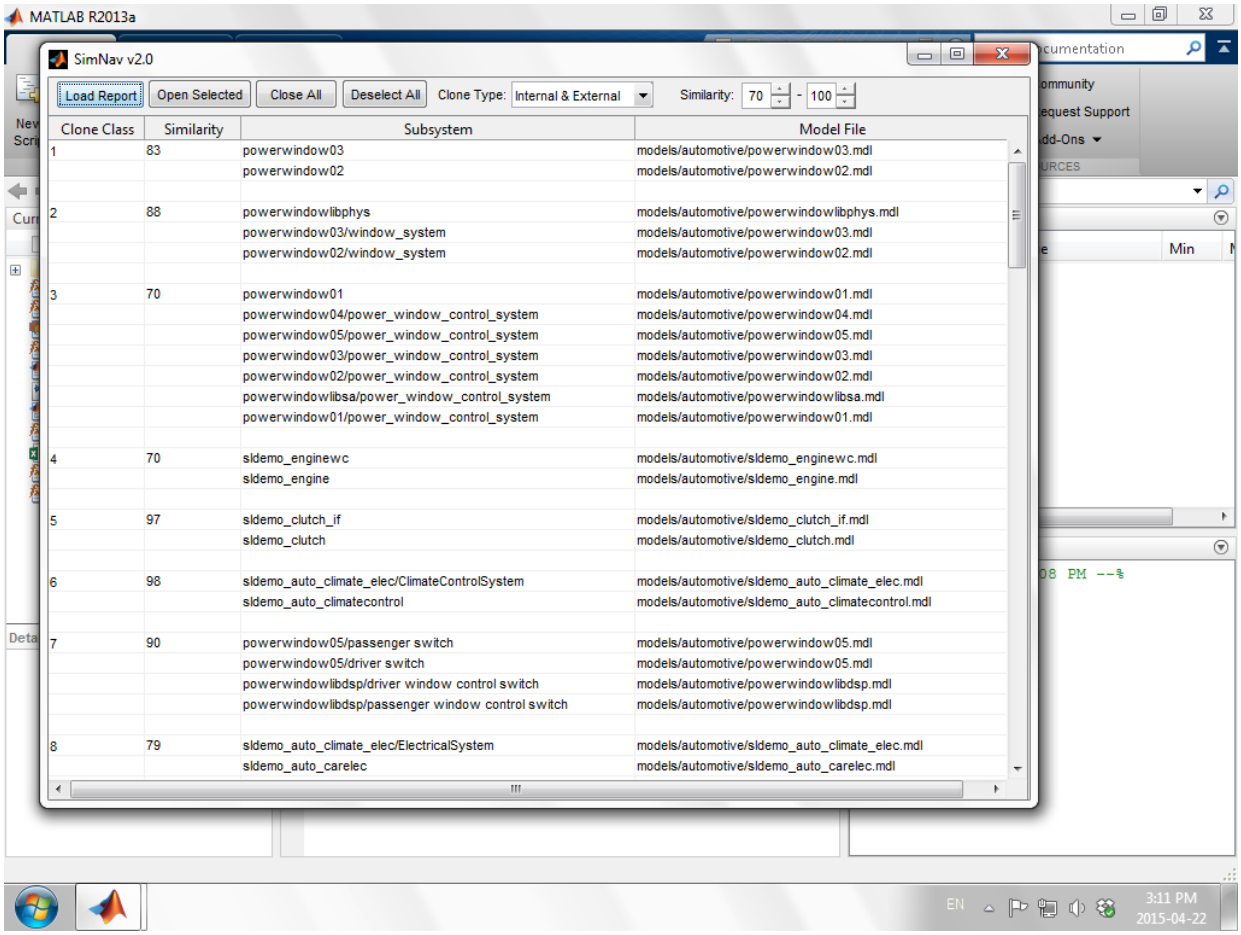

Fig. 3. Main view of *SimNav*

the functionality of choosing any of these reports using a simple file selection box.

After choosing a Simone report, *SimNav* loads up the clone classes and displays them in a table, providing the user with the following attributes: clone class ID, similarity, subsystem name, and model file. An example of a loaded report can be seen in Figure 3.

With this view, the user can examine how many clone classes were detected, the sizes of the clone classes, how similar the clone classes are, and which models and subsystems occur within the clone class.

B. Viewing Clone Class Instances

Once the user has identified the model clones of interest, they may select manually any number of models, or an entire clone class at a time by clicking on the clone class number. The selection of an entire clone class can be seen in Figure 4 for clone class 1.

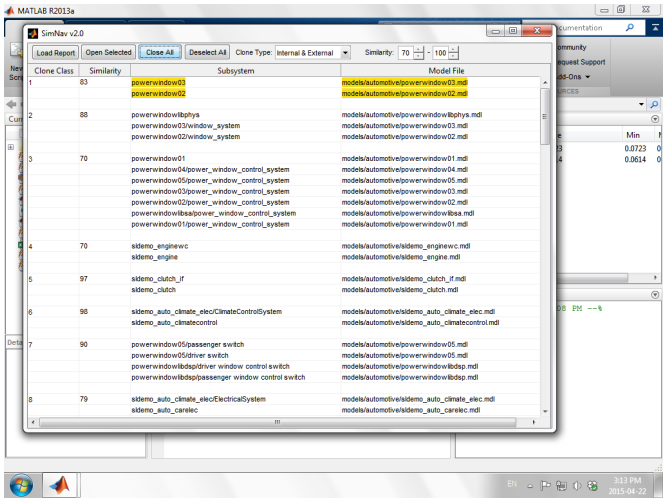

Fig. 4. Selecting all of clone class 1

After clicking on *Open Selected*, the models are opened and the clone instances are tiled on the screen. Figure 5 shows the opening of clone class 1, tiling the two instances side-by-side. Clone class 1 has a similarity of 83%, meaning it is not exact, so a visual inspection of the models can help identify how the models differ. In Figure 5 we see that there is a minor difference at the top level view: the output block on the lower right hand side is different in each model (one is a subsystem and the other is a terminator block).

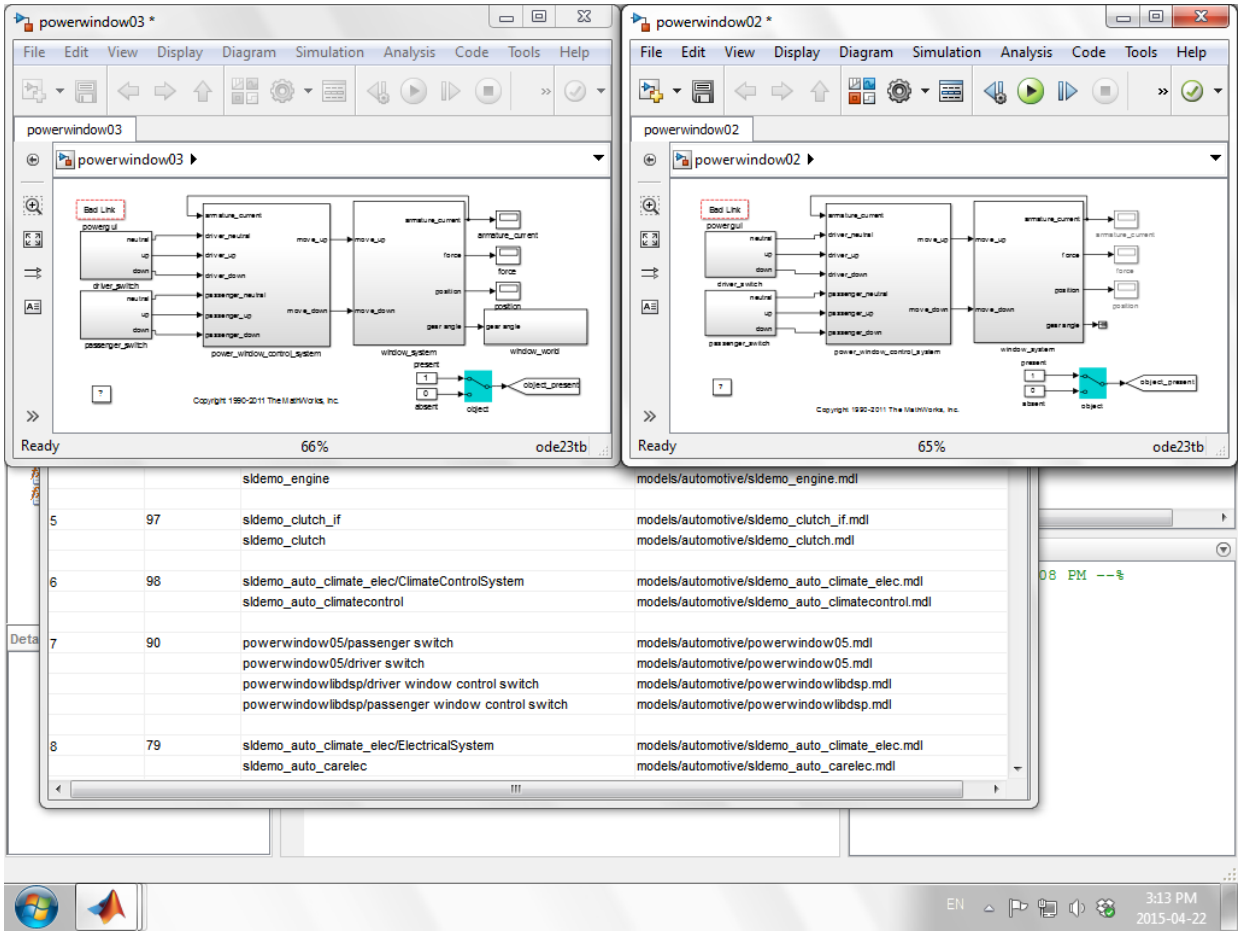

Fig. 5. Opening clone class 1, automatically tiled (2 models)

SimNav is capable of displaying more than two models at a time, which can be seen in Figure 6 which displays the opening of clone class 2, which has three models, and a similarity of 88%. This particular clone class reveals an interesting result, as it found a library block, and two instances of the library block over versions of the model.

Once done examining the models, *SimNav* is capable of closing all of the instances in two different ways: clicking the *Close All* button, or by selecting a new clone class. The first option will close all of the open models, but they will still be selected in the *SimNav* GUI, which can then be deselected using the *Deselect All* button. The second option will automatically close all open models, deselect those models, and select the new clone class.

C. Filtering Results

In addition to simply viewing the clone class results and opening models, *SimNav* provides the user with additional functionality related to filtering the clone class results.

As explained in Section III-B, *SimNav* implements two types of filtering. Using the same model set from the previous use case, and using the *Internal Only* filter, *SimNav* would provide the view in Figure 7 where a number of Clone Classes are filtered out (1,2,4,5,6,8...).

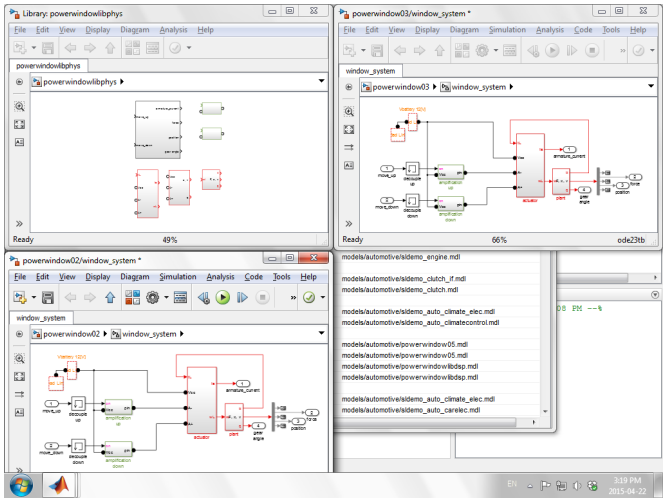

Fig. 6. Opening clone class 2, automatically tiled (3 models)

The other type of filtering available in *SimNav* is based on the Similarity of the clone class, and wanting to remove any clone classes that may not be relevant to the current analysis. Figure 8 shows filtering the models to only display clone

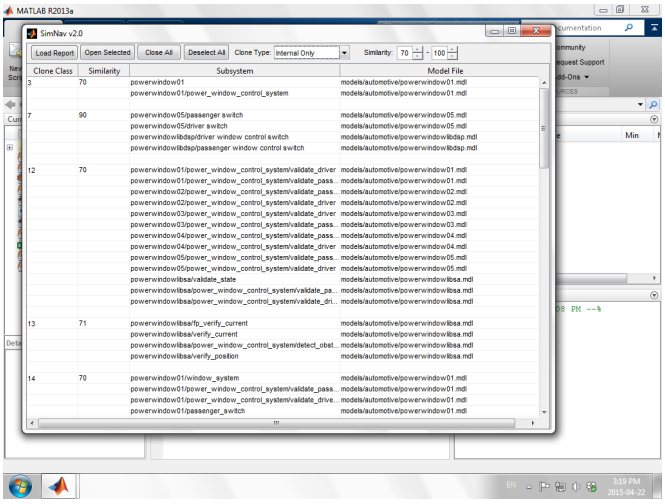

Fig. 7. Filtering results to show only clone classes with internal clones

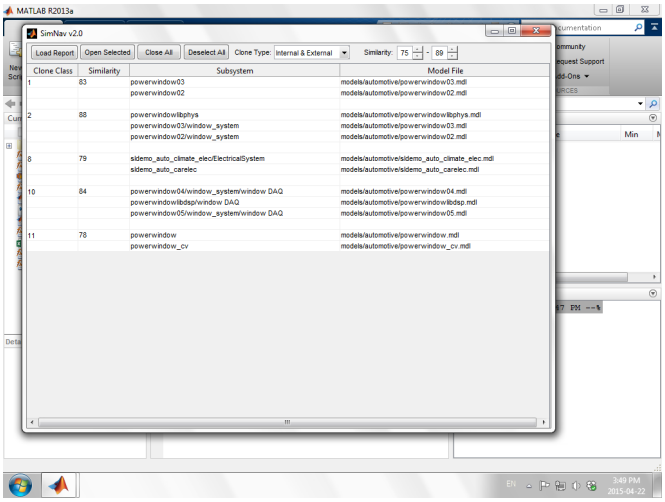

Fig. 8. Filtering results to show only clone classes with similarity between 75% and 89%

classes with similarity between 75% and 89%, which leaves only five clone classes (1, 2, 8, 10, and 11).

V. Future Work

There are still several features that we would like to include in *SimNav* which would increase its usefulness.

The first of these would be the ability to visually identify differences and similarities within a clone class. Currently, users need to manually identify these by looking at instances, but we plan to implement a method of colouring differences and/or similarities, to easily show how models differ. We have begun early work on this feature, which is relatively simple for pairs of models, but increases in complexity as the clone class increases in size.

Another use case for *SimNav* is the ability to create variability models of near-miss clone classes [3]. The ability for SimNav to take a clone class, which is mostly similar, with a few minor differences, and merge them into a single variability model, would allow for the automation of previous work on variability and model product lines, and further improve the maintenance of Simulink models.

VI. Conclusion

SimNav implements the presentation of results from the Simone model clone detector directly in the Simulink environment, a property absent from many other clone detection tools for Simulink models. The ability to view results and open models within Simulink makes *SimNav* an effective tool for navigation of Simulink clone classes. Further the filtering features of *SimNav* add to its usefulness to developers, allowing them to display only clones of interest to them in a particular use case, whether that is clones of a certain type or within a certain similarity range. The industry collaboration involved in the development of *SimNav* has been a huge asset.

SimNav can be downloaded here:
http://cs.queensu.ca/home/eric/SimNav.html

A video demonstrating *SimNav* can be found here:
https://youtu.be/6oPR3U31I8M

Acknowledgments

The authors would like to thank the NECSIS Automotive Research Network, General Motors, and NSERC for their support. Additionally, we would like to thank Kenny-Luc Vuong, Melanie Wightman, and Ron Elbaz for their contributions.

References

[1] Mathworks MATLAB GUI Design. http://www.mathworks.com/discovery/matlab-gui.html. Accessed: 2015-05-21.

[2] M.H. Alalfi, J.R. Cordy, T.R. Dean, M. Stephan, and A. Stevenson. Models are code too: Near-miss clone detection for Simulink models. In *ICSM 2012*, pages 295–304, Sept 2012.

[3] M.H. Alalfi, E.J. Rapos, A. Stevenson, M. Stephan, T.R. Dean, and J.R. Cordy. Semi-automatic identification and representation of subsystem variability in simulink models. In *ICSME 2014*, pages 486–490, 2014.

[4] J.R. Cordy. Submodel pattern extraction for Simulink models. In *SPLC 2013*, pages 7–10, 2013.

[5] F. Deissenboeck, B. Hummel, E. Juergens, M. Pfaehler, and B. Schaetz. Model clone detection in practice. In *IWSC 2010*, pages 57–64, 2010.

[6] E. Juergens, F. Deissenboeck, and B. Hummel. CloneDetective-a workbench for clone detection research. In *ICSE 2009*, pages 603–606, 2009.

[7] H. Petersen. Clone detection in matlab simulink models. *Master's thesis, Technical University of Denmark*, 2012.

[8] N.H. Pham, H.A. Nguyen, T.T Nguyen, J.M. Al-Kofahi, and T.N. Nguyen. Complete and accurate clone detection in graph-based models. In *ICSE 2009*, pages 276–286, 2009.

[9] C.K. Roy and J.R. Cordy. NICAD: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization. In *ICPC 2008*, pages 172–181, 2008.

[10] M. Stephan, M.H. Alalfi, J.R. Cordy, and A. Stevenson. Evolution of model clones in Simulink. In *ME@ MODELS 2013*, pages 40–49, 2013.

[11] M. Stephan and J.R. Cordy. Identifying instances of model design patterns and antipatterns using model clone detection. In *MISE 2015*, pages 48–53, 2015.