DVDMT: A Tool for Dynamic Visualisation of Dependency-based Models

0505049

Abstract

It is very difficult to visualise the relationships between variables and functions in programs developed using procedural and object orientated languages. It is possible to view the static structure of the programs using UML and similar techniques but none of these can show the relationships between variables and functions as a program runs dynamically. This paper looks at how the extra information available within dependency-based technologies can be utilised to provide a dynamic view of the interdependencies between observables. This work has led to the development of a visual dependency viewer users can use to explore and develop executing models. The resulting application shows that it is possible to provide a visual representation of a model when additional information is available to describe the relationships between variables and functions.

[Weighting: Paper 30% / Model 70%]

1 Introduction

In 2007 research was carried out into existing software for visualising software architecture. This looked at the requirements of the users and features which must be provided. The conclusion of the paper stated that "Most tools do reasonably well in static representation. Dynamic representation is another matter, as none of the surveyed tools have support for this key area" (Gallagher, Hatch, & Munro, 2007).

This research looks at the problems in providing dynamic visualisation of executing programs. Dynamic visualisation is the ability to monitor the current state of software as its being executed rather than static visualisation which only allows the source code to be viewed. This paper details some existing attempts at dynamic visualisation tools and the problems with their implementations.

2 Storing Values within Programs

Many programming technologies use *variables* to store the value resulting from some procedural code. This means that the variables themselves are only aware of their current value and not how this value was determined. The value of a variable is usually updated when a function is triggered by some event, for example a user pressing an 'update' button.

An alternative to variables is the use of *observables*. An observable not only stores its current value but also its definition which defines how it is evaluated and thus which, if any, observables it is dependent on. This definition can range from a fixed value to a complex expression involving condition statements, mathematical expressions and function calls. This means that the value of an observable can be automatically re-evaluated when any observable it depends upon is modified.

The use of observables and their dependencies is a core feature of spreadsheet applications in which users can define expressions in cells (the observables), which will be maintained as other cells are updated. Empirical Modelling[1] [EM] also uses this dependency architecture at its core. The primary tool used in EM is EDEN[2] which is a powerful dependency maintenance engine in which observables can be assigned values, formulas, procedures and functions with definitions being defined in definitive scripts which define the EM model, which can be regarded as a program. The information about the observable definitions can be used to produce a graphical visualisation which illustrates the dependencies within a model.

[1] http://www2.warwick.ac.uk/fac/sci/dcs/research/em/
[2] http://www2.warwick.ac.uk/fac/sci/dcs/research/em/software/eden/

3 Problems with Existing Tools

Visualisation of software architecture is often carried out during the process of software design to plan out how the components of a software project should function and interact. Many tools exist to support this visualisation through UML, and other diagrams.

These tools suffer two major problems, however, in helping developers explore the architecture of software. Firstly, none of the tools reviewed currently offer dynamic support to explore a running application. Secondly, the visualisation tools often provide no information about how a variable or function is defined, where the value of a variable is assigned or how these variables and functions truly relate.

4 How can Dependency Modelling and Visualisation Benefit Users?

Using dependency based technologies allows for far more information to be presented to users and also provide simpler means for them to adapt the application. This data can be used to produce a set of directed graphs G = {V,E} to represent the relationships within the model, where V is the set of all observables and E are the dependencies between them.

The possibility of visualising definitive models has already been explored within the EDEN environment in two applications. The first was the Dependency Modelling Tool [DMT] (Wong, 2003) which was actually a Java parser for the definitive scripts, used to define models in EDEN. This, however, suffers a similar problem to existing tools mentioned before in that it was not dynamically linked into EDEN and was therefore a static visualisation model.

The second model is the EDEN Dependency Modelling Tool [EDMT] (Harfield, 2006). The EDMT is dynamic model developed to actually run within the EDEN environment alongside another model. This allows it to maintain an up-to-date view of the current state of the model as its being used. The EDMT model suffers some problems in both usability and design. It does not allow a single path of definitions to be filtered out and their paths explored which is one of the key requirements necessary for software architecture visualisation to be successful (Gallagher, Hatch, & Munro, 2007). Because the model is run alongside the model being viewed, there is a possibility that the observables used to create the EDMT may conflict with observables within the model.

5 The Dynamic Visual Dependency Modelling Tool (DVDMT)

This research has looked at possible solutions to the problems of exploring models dynamically and the resulting application produced is the Dynamic Visual Dependency Modelling Tool [DVDMT]. The DVDMT improves upon the prior visualisation models produced for use with EDEN, such as the EDMT, discussed previously. It looks to meet many of the requirements identified for a successful Software Architecture Visualisation Tool, with the key advantage that it is able to display a dynamic representation of the model as it is running.

The DVDMT is developed to utilise Web EDEN, an extension of the existing EDEN tool, which allows applications to interact with EDEN over a network connection. This allows the DVDMT to access information about an EDEN model while not becoming directly integrated into the model being viewed.

It also provides a range of features to users including the ability to view only those observables which have changed since a previous snapshot was taken. This means that the user can view the impact of a single change on the model. The different types of observables, such as strings, integers, functions and procedures are highlighted in different colours.

The graph can also be presented using a range of different layouts, selectable by the user, and they can also modify the style of nodes and edges from those available. Additionally, to help with navigation of the graph, users can vary the depth of dependency represented. For example, the user may just wish to view observables that are directly dependent upon the active observable, or to view up to ten levels of dependency. To support this, a history feature allows a route to be traced through the dependencies, so that those previously viewed remain on display.

The DVDMT also shows the connected observables from a given observable, rather than representing every observable defined on screen at the same time. This avoids cluttering the graph with excess observables that are not related. Different starting observables can however be selected from a searchable list of all observables.

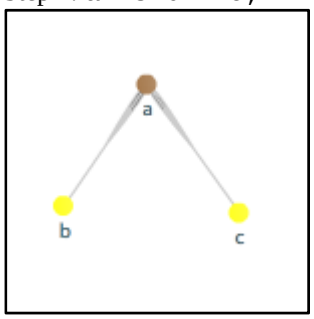
It also provides a number of other features, including the ability to extract more detailed information about observables such as the observables type, definition and current value. This is accompanied by the ability to update and interact with the model by (re)defining observables and dependencies.

Many of the items highlighted above can be seen marked on Figure 2 [see Appendix 2: A-I].

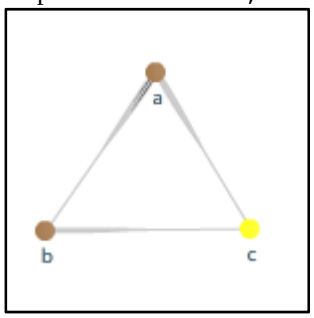
6 The DVDMT in Use

A simple model using the observables *a*, *b* and *c* has been used to illustrate the graphs rendered by the DVDMT. The 3 steps below show the command executed in EDEN and the corresponding graph the DVDMT produces:

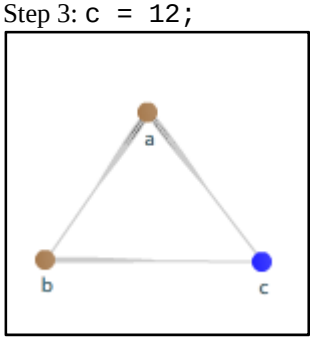
- Step 1: `a is b + c;`



- Step 2: `b is 2 * c;`



- Step 3: `c = 12;`



In the three illustrations it is possible to see that the dependencies between observables. The edge style used in these examples starts with a wider line at the parent observable which narrows towards the dependent observable. In addition the colour coding of observables can be seen to change as the type of each observable is updated.

Some examples of the full DVDMT interface have also been included within Appendix 1. Appendix 3 includes some further examples of the DVDMT displaying various dependencies within the OXO Model (Joy, 1994) using different layout settings.

7 Conclusion

The Dynamic Visual Dependency Modelling Tool shows clearly how dependencies within models can be illustrated to users to provide them with the ability to explore a model and gain a greater understanding of them. It also shows that, by developing programs using a dependency-based modelling technology, it is possible to provide dynamic visualisation of programs.

It is hoped that this work will help users of EDEN but also help increase the awareness of how dependency technologies like EDEN and spreadsheet applications can be used to help benefit users and developers. However, in areas where efficiency is more important than the ability to visualise a program that procedural based languages are more appropriate.

It is intended that the DVDMT will become an accessible option from within Web EDEN Interface and form another key component of the Web EDEN collection of applications. Other potential extensions include the implementation of a faster system of extracting changes to EDEN observables. This would likely take the form of a single list being maintained by the core of the EDEN engine containing the names of all observables added or modified and observables being removed or cleared when removed or a new snapshot is started. Other possibilities include allowing users to update definitions from within the DVDMT and provision of a wizard for creating and modifying observables.

Trying out the DVDMT

Information on how to try out the DVDMT can be found in Appendix 4.

Acknowledgements

I would like to thank Meurig Beynon for his help in determining the direction for this paper. And Fiona Holder for her help in reviewing it.

References

Gallagher, K., Hatch, A., & Munro, M. (2007). *A Framework for Software Architecture Visualisation Assessment.*

Harfield, A. (2006). *Dependency Viewer*. Retrieved May 2008, from http://empublic.dcs.warwick.ac.uk/projects/dmtHarfield2006/

Joy, M. (1994). *OXO*. Retrieved May 2008, from http://empublic.dcs.warwick.ac.uk/projects/oxoJoy1994/

Wong, A. (2003). *The Dependency Modelling Tool*. Retrieved May 2008, from http://empublic.dcs.warwick.ac.uk/projects/dmtWong2003/

Appendix 1

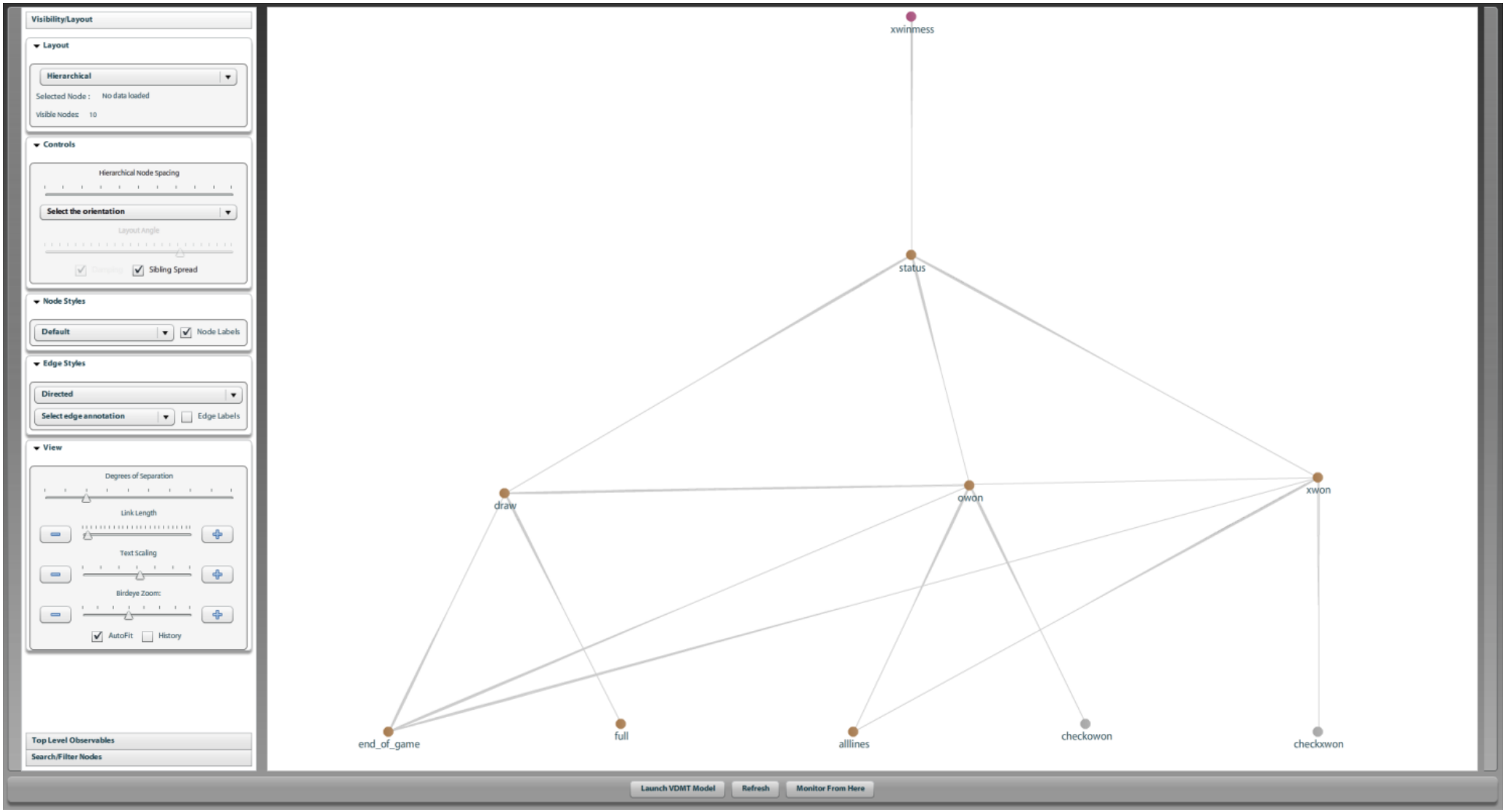
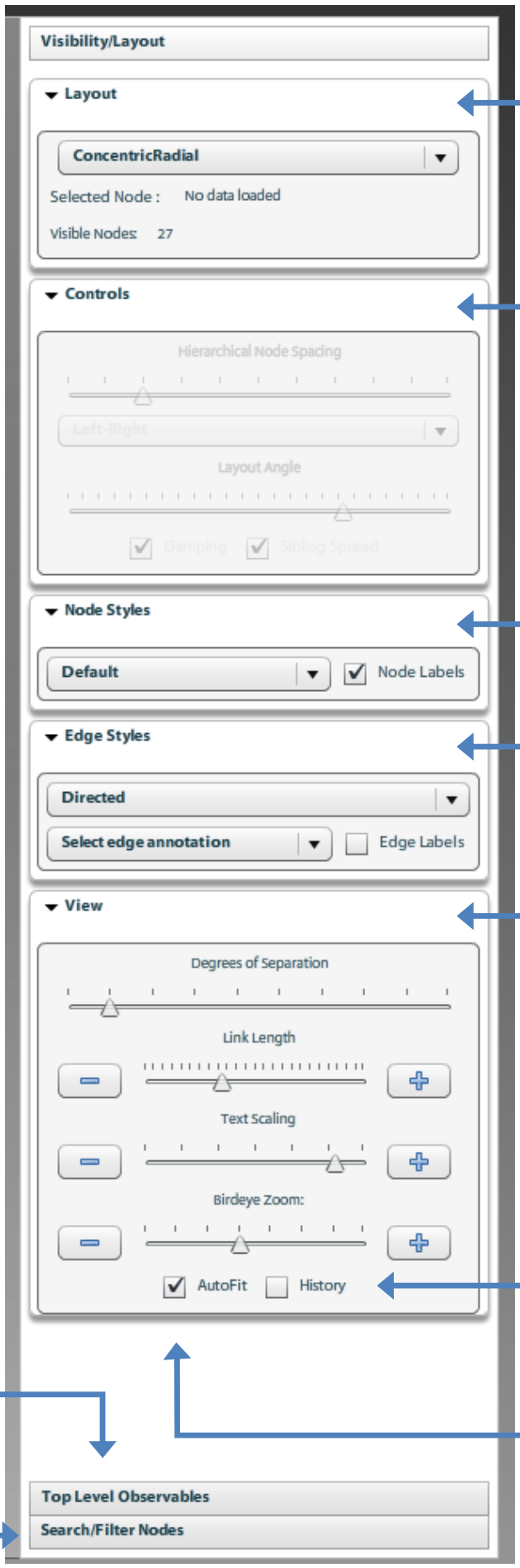


Figure 1: DVDMT interface showing the hierarchical graph layout

Appendix 2



A) Graph Layout Options

- Concentric Radial
- Parental Radial
- Hierarchical
- Force Directed
- Direct Placement

B) Layout Specific Options

C) Node Options

D) Edge Options

E) View Options

F) Graph Exploration History

G) AutoFit to fit graph into available space

H) List all top-level nodes. These are observables on which no other observable is dependent. Although, it may well depend on other observables.

I) Searchable list of all observables

Figure 2: DVDMT Layout Options

Appendix 3

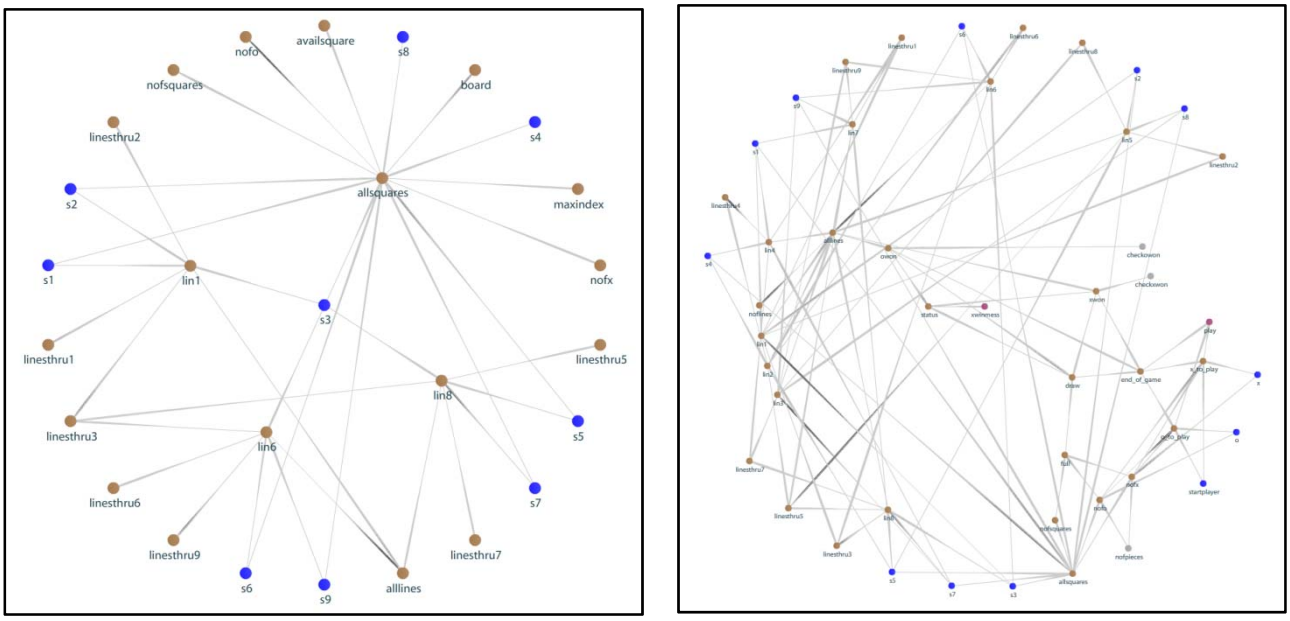


Figure 3: DVDMT showing radial layout options with varying degrees of dependency depth shown.

Appendix 4

The instructions below will help you get started with trying out the Dynamic Visual Dependency Modelling Tool [DVDMT]. These instructions are correct as of the 5th May 2008:

1. Loading up Web EDEN
    a. Go to https://weden1.dcs.warwick.ac.uk/ (Note that is https and not http and you must accept the certificates)
    b. Select to "Try Web EDEN"
    c. Start a new Web EDEN session
    d. Once loaded select from the main menu "Help > Session Sharing"
    e. Take note of the session name and copy the password to your clipboard
    f. Do not close this browser window or navigate from this site!

2. Loading up the DVDMT
    a. Now open a new browser window and go to the site listed in step 1.a
    b. Select to "Try the DVDMT"
    c. Once loaded press the "Refresh Sessions" button
    d. Select the session you identified in step 5 and paste the password into the password box and then press the "Connect to Session" button.
    e. Select the "Load Model" button at the bottom of the screen
    f. If this has all worked out okay then 2 new button should appear at the bottom of the screen

3. Simple Example of the DVDMT
    a. Go to the instance of Web EDEN you opened up in step 1
    b. In the input window type

```
a is b + c;
```

    c. Press the Execute button
    d. Return to the DVDMT
    e. Press the refresh button
    f. Now go back to Web EDEN window and type the following and press execute again

```
b is 2 * c;
```

    g. Return to the DVDMT
    h. From the layout options select "Hierarchical", then press the refresh button.
    i. Finally return to Web EDEN and type the following

```
c = 12;
```

    j. Return the DVDMT and press refresh one last time. You should now be able to select any of the 3 nodes shown and see information about them within the right hand panel.

4. Trying the DVDMT with Other Models
    a. You will want to restart both Web EDEN and the DVDMT before starting this by repeating steps 1 and 2
    b. Now you can go into Web EDEN and select from the main menu "Examples > Quick Launch" and then select one of the models available. A good example model is the "Noughts-and-Crosses" example. So select this from the options available
    c. Now select the execute button to load the model
    d. Return to the DVDMT and select refresh
    e. You can now start exploring the model.

Further Notes

- To view top level nodes which are not dependencies to any other observable select the "Top Level Observables" button from the bottom left control panel. The double click on an observable of interest to focus upon it.

- To view find a specific observable to view select the "Search/Filter nodes" button also found at the bottom of the left control panel. You can then enter the start of the name of the observable of interest within the search box.

- If you want to see what impact later changes make to the model you can select the "Monitor from here" button at the bottom of the screen. Then go into Web EDEN and make an update to an observable. Return to the DVDMT and select Refresh. This will then show all the observables which have in some way been updated / changed definition since you clicked on the "Monitor from here" button.