CS405 Empirical Modelling

MSc Coursework – Comprehension Exercise

Authors:	Judy Palimonka, Clare Donnelly
Model Studied:	ViModes
Date:	November 2012

The goal of this handout is to provide some more information with regards to our understanding of the vi model, in addition to the presentation. This document comprises of the following sections:

1. Introduction

The purpose of the viModes is to enable a modeller to develop a construal of vi text editor. Another goal stated by the author was to help modeller understand scout definitions. The priority for us however was to present observables with respect to vi editor and we did not address the secondary goals of this model.

Vi was an early, console-based text editor designed for Unix systems. It operated in "modes", i.e. the insert mode (where the typed text became part of the document) and the normal mode (where keystrokes were interpreted as commands). In order to move from one mode to another, a certain key combination had to be entered. Vi wasn't as easy and straightforward to use as contemporary text editors and in order to gain the basic understanding of how the application works, users had to refer to the program's manual as well as consult the reference of the key combinations corresponding to various commands.

The purpose of ViModes model is to demonstrate the behaviour of VI editor. With the aid of graphical interface, it lets the modeller observe the immediate impact of the interaction with the artefacts and grasp the idea of working in "modes".

2. The Implementation of EM Principles in The Model

We will first attempt to demonstrate our understanding of construal using simple terms by distinguishing the key observables and the basic dependencies between them. Drawing on the visualisation of the model, we discuss the situation in terms of real-world observables of interest. Then we look more closely into the definitive scripts and outline the set of variables, functions and definitions governing/representing the identified observables, how they are defining dependencies and how procedural actions are triggered in the model.

2.1 General Overview of Construal

The most obvious observables can be spotted at the first sight of the model's visual representation. The observables we have captured are as follows:

Display Console

This part of the panel shows the text entered so far. It also reflects the result of each interaction with the consoles outlined below and helps to track the changes / spot the modes. The input console consists of the related observables such as:

- Text Area - the field with currently edited text
- Cursor - the currently highlighted character in the text
- Mode indicator (the colour of the cursor indicating the current mode)

Reviewing Console

The pink area contains the controls that are accessible only in the "review" mode. The navigational controls (arrows) allow for moving the cursor through the text in all directions. The controls x and dd delete the currently highlighted character or the current line. The controls a and i allow for text insertion (before or after the cursor respectively). The interaction with the last two controls lead to the change of mode to "editing" and in result the editing console to be enabled for input.

Editing Console

The green section contains a visualisation of a keyboard. The interaction with it is visible in the display above. In this case, the interaction does not lead to change of mode. Escape button does however, result in altering the mode to "review".

Escaping Mode Console

Exiting controls such as u, d or vi would lead to change of mode back to "review".

Naturally, each change of mode results in making the corresponding "console" active while blocking all other consoles. What follows, is that the buttons associated with a particular console are blocked along with the console.

This was only a basic and very intuitive overview of the observables in the construal. The model consists of many more smaller observables with a number of dependencies described in the next section.

2.2 Observables, Dependencies and Agents in The Model

This section outlines the list of observable as defined in the scripts attached. In the effort to present the model and its observables clearly, we will list all key observable in the form of tree where the dependants are shown higher (wider margin) and determinant observables are below the definitions relaying on them. Also the attached Dependency Modelling Diagram has the key observables and their dependences presented in quite an accessible and easy to understand manner.

Constants, Observables and Dependencies

- **currentmode** - current mode (reviewing, input, escaping)
- **screen** - the visual representation of the construal, contains input and four consoles
  - **docwin** - input display
    - **textediting** - the content of input display (the string being reviewed/edited)
      - **line1** – text contained in line 1
        - **initline1** – beginning of line 1 (value depends on the content of line 1)
          - **mark1** – sign starting line 1
      - **line2** – text contained in line 2
        - **initline2**
          - **mark2** – sign starting line 2
      - **line3** – text contained in line 3
        - **initline3**
          - **mark3** – sign starting line 3
      - **line4** – text contained in line 4
        - **initline4**
          - **mark4** – sign starting line 3
  - **marks** – a collection of signs starting each line containing marks 1-4
  - **highlight** – rectangular highlighting current character
    - **currentcolour** – colour corresponding to mode
    - **textbox** – a place holder for the text
  - **currline** - currently selected line
  - **colno** - currently selected column
  - **curchar** - currently selected character
  - **reviewmode** – the console containing the controls for review mode
    - **keys** – a set of navigation keys for controlling highlight observable
  - **inputmode** – the console containing the controls for input mode

escapemode – the console containing the controls for escaping mode
inputkeys – a box visualising a keyboard for entering characters
escbutts - all controls belonging to the escaping box
alphstrs – lines of characters in the visual keyboard
  kbtopline – top line
  kbmidline – middle line
  kbotline – bottom line
maxlen – the maximum number of characters

2.3 Dependencies Only

uKey eden dependencies presented in the format: dependent observable ~ determinant observable.

screen ~ highlight, docwin, chmodebutts, keys, reviewmode, inputkeys, escape, inputmode, escbutts, escapemode
linelist ~ line1, line2, line3, line 4
textediting ~ line1, line2, line3, line 4
mark 1 ~ initline1
mark 2 ~ initline2
mark 3 ~ initline3
mark 4 ~ initline4
colno ~ lines, setcolno, maxlen, max, min
currchar ~ colno, lines
highlight ~ currchar, textbox, currentcolour
docwin ~ textediting
currline ~ setcurrline
inputkeys ~ inputalphabet1, inputalphabet2, inputalphabet3, inputnewline
curselchar ~ poidentpr, aphstrs, substr
currentcolour~inserting,reviewing,currentmode

Identifying Agents, Actions and Stimulus-Response Patterns

The modeller is an only agent in this model since he/she is the only source of random changes to the state of observables. The natural interaction with the model is done via controls i.e. buttons provided for each console. Each instance of the interaction with buttons triggers an action and causes the relevant set of observables to change. An action is a procedure which, when called, changes the definition of an observable. The interaction with controls in this model causes changes to the state of *several* observables at once. We can consider the controls as a way to stimulate model while the display is a source of response to stimulus.

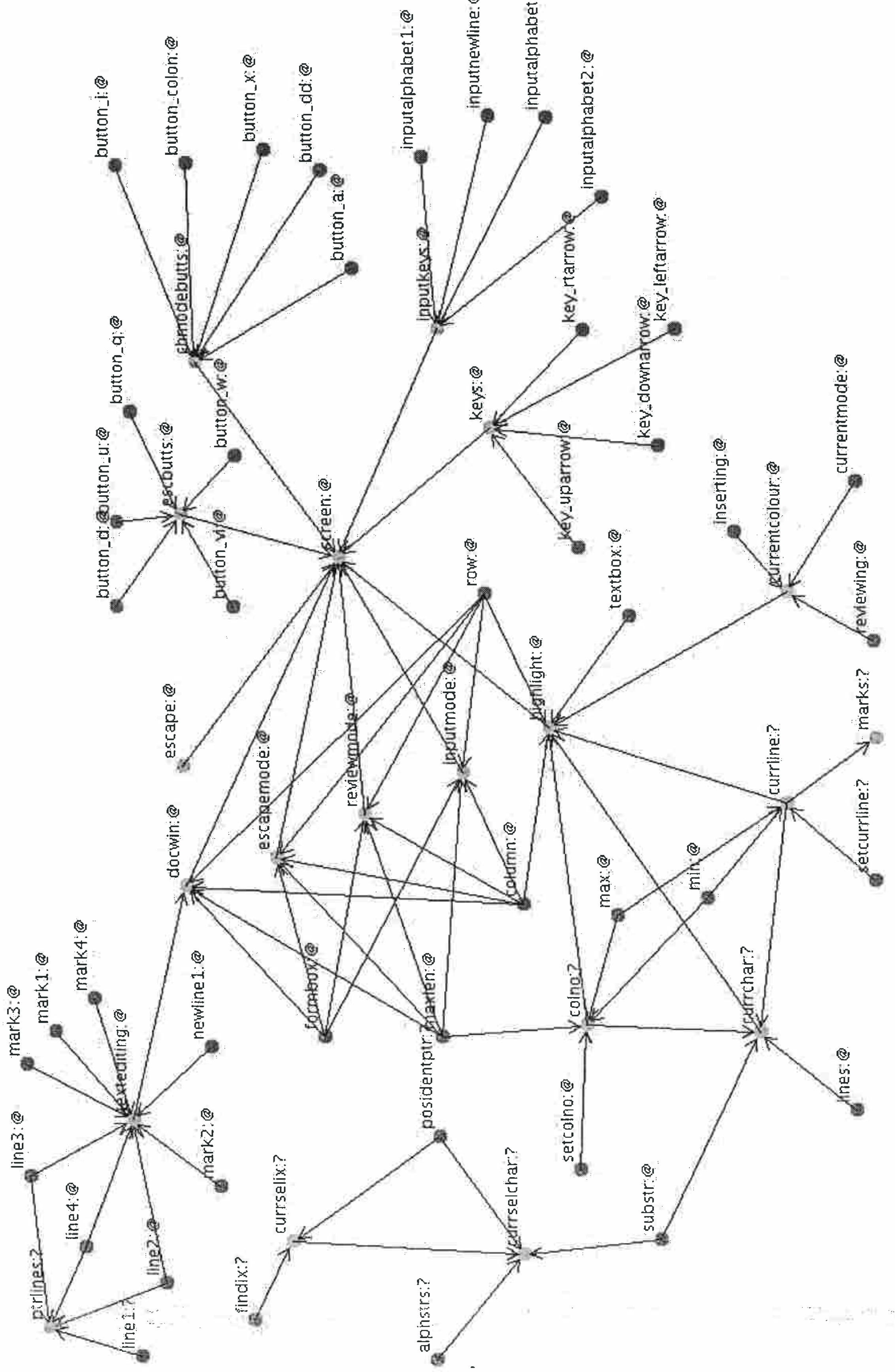
The interaction with navigation buttons results in changes to the status of theu observables **currlline, currchar, curcol** are the controls in the reviewing console:
  key_uparrow – move up
  key_downarrow - move down
  key_leftarrow - move left
  key_rightarrow – move right
The procedures that are called on the interaction with navigational buttons: act_downarrow, act_uparrowact_up, arrow-act_rtarrow, act_leftarrow

An interaction with the controls below changes the status of the observables **currentmode, currchar** and **currentcolour, churrchar**:
  button_colon - causes the **currentmode** to change to escaping
  button_a - append button, causes the **currentmode** to change to reviewing
  button_i - insert button causes the **currentmode** to change to reviewing
  button_d - delete button, changes **currentmode** to reviewing
  button_u - undo button, changes **currentmode** to reviewing
  button_q - q button in the escape console
  escape – a button used to escape the reviewing back to editing
  inputalphabet – a visualisation of a keyboard, makes new input to appear in display
  inputnewline – enters new line display

Procedures involved: presscolonesc,presiinsert, pressainsert, undo, pressdinsert, presviinsert, pressescrev pressxdel, midpos, toppos, botpos

3. Dependency Modelling Diagram

The diagram with the dependencies between observables and agencies is on the next page.



4. The pros, cons and potential of the construal

Pros:

- Most of the functionality works as vi would.
- The button sections and cursor modes are colour coordinated, which makes it seem more likely that the buttons to press in that circumstance are obvious.
- The buttons that change modes are coloured with the mode, this could help a link be made between pressing the button and changing the mode.
- The model is simplified.

Cons:

- The model differs in behaviour from the program vi which may result in modeller being unable to develop a adequate construal in his/her mind.
- It is not immediately obvious how the buttons map to keys (Esc is a key but Vi is not, or may be but isn't marked like that.).
- It may be awkward if a user switches and the mode is not shown by colour. There is no way around it because it is a useful teaching thing just not like the end program.

Potential of construal:

The construal has potential use for letting people play around with the logic of vi without the chance of writing over a file unintentionally. It has also got use in showing he most commonly used features of vi by the modeller. It does this because the most used functions and features tend to be the better understood and once a feature is understood a similar effect can be created in the model. More commonly used features will tend to be the ones thought of first so a model will often show the more thought of an used features rather than the less. It also could be used in showing the modeller's beliefs about the referent though if another person were looking at it this may be unfair unless the above point about familiarity is taken into account.

5. Conclusion

The vimodes is a model of the text editor vi. Vi has a great many features including several control modes. These are modelled in this construal. The construal has many possible uses including showing things that the modeller believes about vi, this could be used when learning about the tool. It has a number of observables and dependencies to show parts of the text editor as well as modelling a keyboard and other input keys. The construal has advantages in providing a colour system to aid in learning the modes as the keys in the program are available in each mode so this helps the user know what is expected. It could be improved in the sense of an aid to learning by making it follow more strictly the behaviours in vim.

Foundations of Technical Analysis: A Summary

Adam Riley
Deparment of Computer Science
University of Warwick

November 7, 2012

1 Handout

The Game: Nim

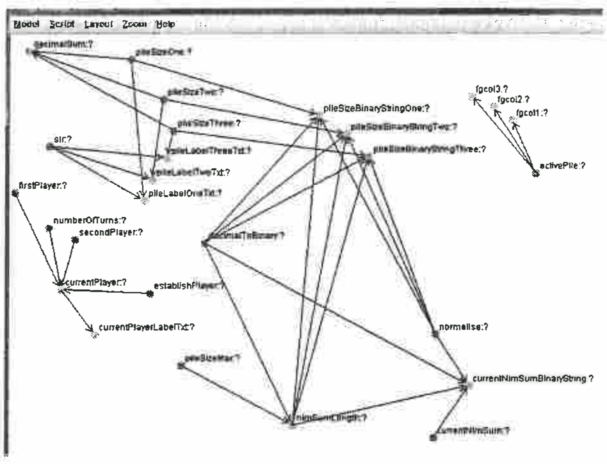
Premise: Players take turns to remove a number of objects from one of several piles.

Win Condition: Take the last object in the game

Strategy: Nim Sum
Ensure that the binary sum of all piles is 0.

By ensuring that the Nim-Sum is 0, the player can guarentee that they can replicate any move made by the opposing player. In the final case,

Below is a model of the Nim construal dependencies:



Subject: Tomorrow's presentation session
From: Meurig Beynon <wmb@dcs.warwick.ac.uk>
Date: 07/11/2012 17:12
To: Steve Russ <sbr@dcs.warwick.ac.uk>

```
Dear Steve

I attach the index file for tomorrow's session.

Adam Riley - 25 observables / 5 slides - NIM

Huang Long Tran - 60+ obs / 18 slides - NIM

Clare Donnelly + Judy Palimonka  (Vimodes)
6 slides + handout + a list of a few undocumented obs

David Walton (Cricket) 38 obs + 17 slides (starts on Slide 10)

Hope this is all you need to know for background.

Bo and Hui were most helpful with this last year - I'll look out a copy of the
sheet for marking we drew up (based on his input) last year.

Best wishes

Meurig
```

— index.txt —

```
1264707 Hoang Long Tran
0902814 Adam Riley
1266015 Judy Palimonka
0927324 Clare Donnelly
0903457 David Walton
```

— Attachments: —

index.txt 115 bytes

grasp of construal
grasp of EM principles / tools
orgn & commn
h/o

2 awarded

A	L	C+J		D
5	5	5	5	5
2	~~5~~ 4	4	4	~~5~~ 4
2	~~5~~ 4	3	5	5
9	~~15~~ 13	12	14	~~15~~ 14

Long　　　Adam
C+J　　　Dave