Cheating Literacy: The Limitations of Simulated Classroom Discourse in Educational Software for Children

Marion Walton
Centre for Film and Media Studies, University of Cape Town, South Africa

This paper presents a multimodal discourse analysis of children using ‘drill-and-practice’ literacy software at a primary school in the Western Cape, South Africa. The children’s interactions with the software are analysed. The software has serious limitations which arise from the global political economy of the educational software industry. The package was structured around the UK National Curriculum’s standardised literacy testing, and then adapted or ‘localised’ for use in South Africa. In the localisation process, details of content and language are customised, but the coded structure of the package (together with its educational assumptions) remains essentially unchanged. The children’s interactions with the localised program are analysed as a simulation of classroom discourse. Despite the obvious limitations of the software, the study shows the children constructing their own contextual meanings from the rules of the package, and learning to interact with them as a rule-governed text. Their troubleshooting and cheating exploits are a source of pleasure to them, as they focus on the software’s game-like economy of scores and marks.

doi: 10.2167/le747.0

Keywords: drill-and-practice literacy software, local, global, multimodal analysis

Introduction

In well-resourced societies, the literacies of the ‘new communication order’ have become essential to communication (Kress & Van Leeuwen, 1996; Snyder, 2001; Street, 1998). Consequently, literacy teachers have adapted their curricula to include networked communication such as chat and online discussion (see for example, Goodfellow, 2004 and discussions in Snyder, 1997 and Snyder & Beavis, 2004), digital media and games (Beavis, 2004; Buckingham, 2003), and experiments with emerging genres such as blogs, wikis and podcasting (Richardson, 2006). For the majority of the world, however, these ‘new literacies’ are exotic practices, sustained by resources and leisure which are simply not available to most people. In South Africa, Internet access is still confined to a small minority, while burgeoning levels of print literacy and growing demand for text messaging and mobile telephony would probably constitute the ‘new literacies’ of the majority.

Until very recently a large majority of South African children had no access to computers at school. Since the country’s first democratic elections in 1994, schools have been under pressure to provide more equitable access to computers.
and other ICTs. The Western Cape province (the focus of this study) has embarked on a process of rapid deployment of computers to all state schools (Dugmore, 2004). Literacy and numeracy development have been targeted as priority areas for these new computer facilities in Western Cape primary schools. Imported software is installed in the laboratories and provides many hours of self-contained ‘drill-and-practice’ literacy lessons. These lessons emphasise grammatical ‘correctness’ rather than meaning-making. This approach reinforces the prevailing notions of literacy teaching in township schools in the Western Cape, where literacy teaching is most often a drill-and-practice activity, and where few teachers adopt an approach which encourages children’s active engagement in meaning-making (Prinsloo, 2005: 7).

For the study reported in this paper, a series of pilot interviews and classroom observations were conducted with the aim of investigating how literacy teaching and learning in primary schools in South Africa’s Western Cape province has adapted to the recent provision of new computer laboratories. The study develops a social semiotic framework for children’s use of software. Interviews and observations provide the context for a multimodal discourse analysis of children using educational software during an Afrikaans literacy lesson at Mountainside Primary, a township school near Athlone in the Western Cape. The following questions guided the research:

1. How do children create meaning from software interfaces, and what interests guide their use of educational software?
2. How does literacy software encode classroom discourse patterns?

**Software as representational mode**

The drill-and-practice exercises of the ‘first generation’ of educational software can be seen as a representational mode which simulates the interaction patterns of classroom discourse through rule-governed sequencing of images, text, audio and sometimes video and animation. Children using such software interpret and negotiate the simulated discourse exchanges as they work through the lessons. But what does it mean to say that software, like photography or speech, is a representational mode?

Jewitt (2004: 184) explains that from a social semiotic point of view, semiotic modes such as speech, writing and images are ‘technologies of representation’. These semiotic modes have been moulded through social use into evolving ensembles of representational resources. By contrast, communicational media such as printed paper or television are ‘technologies of dissemination’ which carry semiotic messages. Jewitt interprets the digital medium as a technology of dissemination. She points out that unlike other technologies of dissemination, the digital medium is also a medium of production (Jewitt, 2004: 184). While digital media allow the combination of different semiotic modes, I would argue that the source code and interfaces of software are more than a medium. They can also be considered a semiotic mode, since they enable and constrain representational possibilities—both for the programmer and for the end user. Thus, computer languages and user interfaces have developed as new representational modes. Educational ‘drill and skill’ programs, as examples of the software mode, are the focus of this paper.
According to Manovich (2001), the affordances of the coded paradigms and syntagms of software and new media constitute a new representational ‘language’. This insight can be combined with social semiotic theories of multimodality to argue that the constraints and possibilities for interaction spelled out in a program’s source code are hidden representational resources, which govern the user’s production of a mediated experience. These resources must be explored and understood by users in order for them to use the software. Acting within the limits set by the code, users are able to produce a new text, which is assembled from components provided in the software. Such components display great variety – they can be the millions of colours and individual pixels in an image editing program such as Adobe Photoshop, or the more limited options and strictly controlled sequencing of multiple choice questions in educational software.

From this perspective, then, the characteristic interactions encoded by educational software can be understood as a semiotic mode which marshals interactive resources for the automation of traditional educational and assessment practices. ‘First generation’ educational software often simulates classroom discourse through a dialogue-like pattern of interaction. Such educational software creates a representation of the learners and their learning by evaluating answers and summarising them in a score or grade. Other digital semiotic modes draw on different modal resources – for example, the characteristic interaction patterns of three-dimensional (3D) computer games simulate movement through a 3D space.

**Political economy of educational software**

The educational software discussed in this paper is built around a model of language learning which is not dissimilar from local practices and discourses of literacy in township schools in the Western Cape. It is also a product of the United Kingdom’s National Curriculum, which has been adapted and ‘localised’ for three South African languages. Buckingham and Scanlon’s (2005) analysis of the political economy of educational software in the UK media industry helps to explain the origins of such software. They discuss the broader field of educational publishing as a media industry in the United Kingdom, showing how the trends within educational publishing have affected the business of developing educational software.

They show that, like other media industries, educational publishing is characterised by heavily concentrated patterns of ownership and a highly competitive globalised market. In the UK industry, educational publishing is primarily geared towards the requirements of national educational curricula. These trends are intensified in the production of educational software, since the retail market is smaller and software development is an expensive business. As Buckingham and Scanlon (2005) observe, the market segment for educational software in the United Kingdom is now simply labelled ‘National Curriculum’.

A popular marketing strategy presents sets of exercises that test ‘basic skills’ (as defined by the ‘back to basics’ movement and national standardised tests); ‘The market seems heavily dominated by the imperatives of national testing’ (Buckingham & Scanlon, 2005: 42). This approach lends itself to the development
of generic sets of exercises and activities. The trend towards such decontextualised activities is intensified because, from the perspective of the software developers, profitable educational software titles need to be generic enough to be easily exported to other contexts.

Localisation is a software development practice which allows a piece of software to be adapted so that it can be more broadly marketed. Although the basic coded structures and functionality remain the same, language and other local details are switched to suit a new target market. The process of localisation can be preceded by ‘internationalisation’ (where developers attempt to remove culture-specific elements of a package). ‘Globalisation’ is another strategy, which attempts to cater for a global audience while remaining unmarked by its local origins.

By bundling a set of educational materials and installing them in all schools along with computer laboratories, the Western Cape Education Department effectively created a viable market for educational software producers. The software package used at Mountainside Primary was produced in the United Kingdom by educational publisher Sherston, where it was originally developed to meet the requirements of the UK National Curriculum. The software was then ‘localised’ under license to the UK company, by a South African firm, which translated the English content into Afrikaans, Xhosa and Zulu, and mapped the word- and sentence-based activities onto their equivalents in the local primary school curriculum.

Classroom talk and software

Researchers who have studied children talking as they work at computers have shown that the use of software generates new interesting variants of classroom discourse. Classroom discourse, like the institutional discourse of interviews and lawcourts, is characterised by a distinct imbalance in power – in this case an imbalance between the teacher and the class. Sinclair and Coulthard (1975) proposed a model for classroom talk which described exchanges between teachers and pupils according to the following stages:

1. **Initiation**: The teacher introduces a topic to the class.
2. **Response**: A pupil or more than one pupil from the class respond to the teacher’s initiation.
3. **Feedback** (or follow-up): The teacher responds to and usually evaluates the pupil’s response.

Such an exchange, also referred to as ‘triadic discourse’ (Lemke, 1990; Wells, 1999) or ‘recitation’ (Alexander, 2005), would be considered unacceptably rude between two adults. Still, it is dominant in most classrooms and is the pattern encoded in most educational software.

The characteristic Initiation–Response–Follow-Up (IRF) pattern is encoded in the form of multiple choice questions and computer-based exercises:

Discourse accompanying highly structured programs conforms well to the IRF (Initiation, Response, Follow-up) structure identified by Sinclair and Coulthard in teacher-centred classrooms, with the computer often taking the initiating role. (Fisher, 1997: 81)
The 'closed' questioning style in extreme versions of triadic discourse can work to shut down all dialogic interchange (Wells, 1999), and this is particularly stifling if teachers are not open to unexpected answers and do not encourage children to provide justification and more information, but rather ask a question and then wait for the pupils to read their minds. The coded structures of educational software provide an extreme version of this pattern of discourse.

Fisher’s model (whereby the computer plays an initiating role and children choose how to respond) does not really account adequately for all the discursive effects of triadic discourse in software. Educational software embodies a 'closed' approach to questioning because, if the software is to provide feedback, permissible responses to the initiation move must be predicted and classified in advance. So, in other words, the computer takes the initiating role and also offers the children a set of permissible responses from which they must select. These responses are then evaluated in the feedback. Such software is thus a simulation of triadic discourse. Where this simulation breaks down (as it often does) the results of careless automation create a text where meaning is a problem – cohesive ties do not help the listener construct coherence. Similar phenomena are found elsewhere in language, such as in the speech of very young children, or in bad translations, and are described as 'non-text' by Halliday and Hasan (1976: 23–24).

Automated cohesion in software

Cameron cites a famous example by Harvey Sacks (1972) to describe how language users interpret sequences of utterances as a meaningful whole:

The baby cried.
The mommy picked it up.

To understand this as a sequence of talk, the listener needs to comprehend the cohesive tie which the speaker used to indicate that the two sentences are connected. The word ‘it’ refers back to the baby in the previous sentence. Beyond this tie, the listener probably also needs to have quite a bit of contextual knowledge of a world in which mommies take care of babies. As Cameron (2001: 11–12) argues, ‘we make sense of discourse partly by making guesses based on knowledge about the world’. In all verbal exchanges, speakers create cohesive ties which refer to the previous discourse and to its context: ‘where the interpretation of any item in the discourse requires making reference to some other item in the discourse, there is cohesion’ (Halliday & Hasan, 1976: 11).

Similar processes apply to cohesion in software. The task of designing an educational software exercise often involves designing a framework to be filled by a package of variable media. Cohesion is planned both within individual screens, between sequential screens and within the exchanges of the planned human–computer interaction. Since the design must cater for variable elements, the cohesion needs to try to encompass as many of the user’s projected responses as possible. Cohesive relations always have their origin in specific cultures of use, and despite practices of ‘localisation’, these are exported to other contexts as the functionality of the software. By its variable nature, software tries to cater
for a wide range of future communicative contexts, which may be predicted to some extent, but are essentially unknown.

Although the designer codes the automated cohesion within a software text, the user interprets and uses it and is responsible for making meaning or building coherence through interacting with it. The automated cohesion characteristic of software simulates cohesive text, but users often battle to make coherent connections between the operations of the software text and their own communicative contexts and needs, which were unknown to the designer. The inability of designers to create cohesion with unpredicted future actions of the user often leads to the creation of meaningless ‘non-text’.

Methodology

The research was conducted in a township near Athlone on the Cape Flats. There were 40 children in the class, most of whom lived in the lower middle and working class housing in the area around the school. The lesson reported here was a half-hour session dedicated to an Afrikaans literacy class, held in the computer laboratory. All the children spoke English as a first language, but they were all bilingual with very different levels of fluency in Afrikaans. The children were all in Grade 6 (11–12 years old).

This study was the pilot phase of a larger ongoing research project, and aimed to develop a suitable methodology for multimodal discourse analysis of software in use. Literacy lessons in the Mountainside Primary computer laboratory were observed while children used drill-and-practice software. This paper presents a detailed analysis of selected interactions with the software during one of these lessons. Children’s interactions were recorded as they worked individually and in pairs, completing exercises that tested their knowledge of Afrikaans grammar and punctuation.

I collected the data as I moved around the computer laboratory with a notebook and a digital camera that was capable of taking both stills and video footage. Pairs of children and individuals using the software were recorded. The video footage was transcribed and collated with my notes and photographs of activity on the screen. As this was a pilot study, the class was not accustomed to my presence nor that of a camera. The camera in particular was somewhat disruptive as the children did not have much exposure to digital cameras, and wanted to pose and look at their pictures and recordings on the LCD. During the observations, children would occasionally engage me in dialogue and ask for assistance with an exercise.

In the transcripts selected for analysis, the children and the teacher are referred to by pseudonyms. Mr Jacobs was the teacher. A detailed transcription and analysis is provided of a recording of Stevie, who interacted with the multimodal text of a punctuation exercise, and called on his neighbour Linda to help him troubleshoot a problem he experienced with the coded structures of the software text. Also provided is an account from another student, Brenda, who explained her use of ‘cheat’ strategies to be able to display mastery of an exercise that she did not understand and to draw attention to the shortcomings of the software scoring system.
The recordings were transcribed without attempting to capture pronunciation, since the focus of the analysis was on the interaction with the software, and on the meaning of the interactions. Question marks and exclamation marks were added to indicate question intonation and emphasis. Actions and gestures which contributed to the meaning of the interaction were transcribed as stage directions and enclosed in square brackets [ ]. Screen capture videos of the software were not used; transcription relied instead on video data, still photographs of the screen and my field notes.

**Multimodal discourse analysis**

Kress *et al.* (2001) pioneered multimodal approaches to discourse analysis. This approach recognises that teaching and learning (and other forms of communication) are intricate multimodal processes where spatial configuration, action, gesture, and visual and linguistic resources work together in a multimodal ensemble. This study adapts this approach to document children’s interactions during the lesson at Mountainside Primary.

As Kress *et al.* point out, the complexity of multimodal communication entails significant complexities which have implications for observation, transcription and analysis:

> A multimodal approach can turn what goes on in the classroom into an implausibly intricate and complex event which seems beyond the capabilities of any human brain to manage. (2001: 17)

Multimodal discourse analysis poses particular challenges when applied to a primary school lesson in a computer laboratory, given the sheer amount of data that can be generated. Thus, the method inevitably involves a process of selection. In addition to the complexities of recording normal classroom interaction (as detailed by Kress *et al.*, 2001), the children’s interactions with the software must be recorded. Finally, it is also helpful to document the software’s coded paradigmatic and syntagmatic structures.

**Transcribing children’s interactions with software**

Although previous studies have found that the computer takes an initiating role in the exchanges of children working at computers, researchers, such as Fisher (1997) and Wegerif and Scrimshaw (1997) do not always include software output responses in their transcripts, choosing rather to paraphrase the computer output along with the children’s actions and gestures.

In this study the transcription attempts to reflect the simulated dialogue structure by representing software messages as turns in a dialogue. In the literacy package these turns are often sound files played to simulate verbal dialogue or textual communication via a ‘dialog box’ (to use the interface metaphor of the software). To represent the unfolding of the interaction in time, the conventions of transcribing human–human interaction are used and adapted to represent human–machine interactions. The transcription is coded according to the IRF structure which the software simulated. A separate coding scheme was developed to analyse students’ troubleshooting strategies as they attempted to solve cohesive problems that arose as they interacted with the software.
Children’s gestures were transcribed as stage directions in square brackets, particularly gestures in which the children drew attention to something on the screen. Visual and aural qualities of the computer response are also presented in square brackets. Responses from the software program are included in the transcript in italics. Since the software is in Afrikaans, an English translation is provided in parentheses.

Discourse analytical methods were applied to analyse the transcripts. The study aimed to examine and represent the children’s meaningmaking processes as they interacted with the software, and to look for patterns and repeated strategies in their meaningmaking activities. Patterns of language use, grammar and syntax were not analysed (Mills, 1997: 135).

Interpreting Software and Troubleshooting Non-text

Stevie was observed as he worked on a grammar exercise. Unlike most of the other children in his class, he worked on his own, and wore headphones so that he could listen to audio instructions and feedback as he worked on the exercises. Linda was sitting at the computer next to him where she was working through the same set of exercises with a partner.

<table>
<thead>
<tr>
<th>#</th>
<th>Transcription</th>
<th>Simulated discourse</th>
<th>Troubleshooting</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Software:</strong> [The screen displays ‘Grade 7: Punctuation revision’. The rest of the text instructs Stevie to insert commas into the sentences of a paragraph.]</td>
<td>Initiation</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Stevie:</strong> [Reads the text on the screen carefully, moving his lips as he reads. He uses the cursor as an aid to reading, following the text on the screen with the cursor.]</td>
<td>Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Stevie:</strong> [Adds six full-stops to the paragraph and presses the ‘Is dit reg’ (Is it right?) button]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>Software:</strong> ‘Oops, jy het nog nie alles nie’ (Oops, you haven’t found everything yet.) [A popup displays ‘jy het 0 uit 6 punte in hierdie aktiwiteit’ (You have 0 out of 6 points in this activity).]</td>
<td>Follow-up &amp; Initiation</td>
<td>Problem – non-text</td>
</tr>
<tr>
<td>---</td>
<td>---------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>4</td>
<td>[presses ‘OK’ button]</td>
<td></td>
<td>[Turns to Researcher and hands her the earphones, pointing at the screen]</td>
</tr>
<tr>
<td>5</td>
<td>Stevie: [presses ‘OK’ button]</td>
<td>Responds</td>
<td>Researcher: [Puts on the earphones and tries to find the missing commas.]</td>
</tr>
<tr>
<td>6</td>
<td>Stevie: [Turns to Researcher and hands her the earphones, pointing at the screen]</td>
<td>Identifies problem</td>
<td>Researcher: [Puts on the earphones and tries to find the missing commas.]</td>
</tr>
<tr>
<td>7</td>
<td>Stevie: [presses the ‘Is dit reg’ (Is it right?) button so that Researcher can hear the program’s negative feedback]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Software: ‘Oops, jy het nog nie alles nie’ (Oops, you haven’t found everything yet.)</td>
<td>Follow-up &amp; Initiation</td>
<td>Problem - non-text</td>
</tr>
<tr>
<td>9</td>
<td>Linda: [points at Stevie’s screen]</td>
<td>Debugging: hypothesis</td>
<td>You used fullstops not commas, man.</td>
</tr>
<tr>
<td>10</td>
<td>Stevie: [presses the ‘OK’ button]</td>
<td>Denies hypothesis</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Stevie: ‘No’</td>
<td>Debugging: testing hypothesis</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Stevie: [He looks closely at both screen and keyboard and then replaces all the fullstops with commas. He presses the ‘Is dit reg’ (Is it right?) button.]</td>
<td>Debugging: testing hypothesis</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Software: ‘Oops, jy het nog nie alles nie’ (Oops, you haven’t found everything yet.)</td>
<td>Follow-up &amp; Initiation</td>
<td>Problem - non-text</td>
</tr>
<tr>
<td>14</td>
<td>Stevie: [Turns to Linda and touches her shoulder]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Linda: [Takes the mouse from Stevie and clicks ‘Probeer Weer’ (Try Again) to restart the exercise]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In turns 1–3, Stevie participates in the simulated discourse with the program. The program instructs him to add commas to a paragraph. He responds to the initiation move by adding what he thinks are commas, but he receives negative aural feedback, which translates as ‘Oops, you haven’t found everything yet’ and receives a score of 0/6.

At this point the simulated discourse breaks down, because the response creates a problem of coherence for Stevie, since turn 4 is ‘non-text’. The program responds ‘Oops, you haven’t found everything yet’. The negative feedback and low score are based on a simple count of the number of commas inserted, rather than any awareness of what Stevie has been doing. The feedback is non-text because Stevie has found all the places where commas should be inserted in the paragraph, but the program cannot recognise this. Consequently, it refers to Stevie’s actions in the previous turn in a way which breaks the cohesion of the discourse—the word ‘everything’ refers anaphorically to Stevie’s commas in the previous response, but this does not cohere with Stevie’s experience. The feedback is confusing to Stevie, because he has identified all the places in the paragraph where commas are required, but he has used the wrong key on the keyboard. His response to Linda in turn 10 suggests that this mistake arose from an inadvertent ‘typo’ rather than from lack of familiarity with punctuation.

From turn 4 onwards, Stevie is no longer engaging with the simulated discourse on its own terms. Nonetheless, the discourse can only proceed if he accepts the verdict of the program by pressing the ‘OK’ button. He presses OK, but his subsequent comment to me indicates that he is not in agreement with the feedback and has not in fact accepted the program’s implicit instruction to insert more commas.
Instead of responding to the program’s initiation move and continuing the simulated discourse by adding more commas, or using the ‘tip’ button to get assistance from the program, Stevie appeals to me as the nearest available adult and authority figure in the classroom. His comment in turn 5 (‘They say I got 0 out of 6, Miss’) might be a simple request for clarification, or it might be an appeal against the authority of the program. In either case, he is identifying a problem within the simulation which he needs to troubleshoot.

There are several textual cues here indicating that Stevie steps out of the simulated conversation with the program and questions its authority. First, he chooses to refer to the program as ‘they’. This is an indication that he is not engaging with the simulated discourse, since the pronoun ‘they’ refers to an anonymous plural subject (possibly the program developers). It does not refer to the animated cartoon character who supposedly addresses the user in the software’s interface. It is also significant that Stevie says, ‘They say I got 0 out of 6’ rather than ‘I didn’t find all the commas’ or ‘I got 0 out of 6’. Stevie has not accepted the program’s verdict, and he is more interested in appealing his score than in the task of finding missing commas.

When responding to Stevie’s appeal, I initially took the automated feedback at face value. I assumed that Stevie had not found all the commas—in other words I accepted the program’s diagnosis rather than Stevie’s identification of a problem. The repeated feedback in turn 8 is another example of non-text. First, the program’s ‘you’ lacks exophoric cohesion because the listener is now a different person—the researcher rather than Stevie. Second, the expression of surprise ‘Oops’ is also out of place in a repeated version of the feedback.

In turn 9, Linda overhears Stevie’s comments and offers a debugging suggestion (with some condescension), pointing out what I had not noticed—that Stevie had used full stops rather than commas. Linda’s suggestion could be viewed as a hypothesis in the troubleshooting sequence.

Stevie initially denied that he had used the wrong punctuation mark, then looked closely at both screen and keyboard, and realised that Linda’s diagnosis was correct. He changed his answer, replacing all the full stops with commas. He tested the hypothesis but received the same feedback, which now lacked all cohesion with the display on the screen, and with the previous exchanges in the discourse, since Stevie had in fact inserted all the commas in the right places. The program did not accept his corrections until Linda helped him to restart the exercise from the beginning. This may have been a usability problem with the software.

Stevie inserted all the commas for a third time and was awarded full marks for the exercise, since all previous variables had been reset when Linda helped him restart the exercise. He passed me the earphones so that I could also listen to the positive feedback, which translates as ‘Correct. You got it right the very first time’. Because Linda had restarted the exercise, the program had no record of Stevie’s two previous attempts, and so praised him for completing it correctly first time around. Stevie’s response suggests that he was now happy to step back into the simulated discourse and to accept the program’s praise.

The software was unable to interpret Stevie’s input as a human teacher would have (and as Linda had been able to). In this case, ‘drill-and-practice’ meant that Stevie had to ‘drill-and-practice’ a skill which he had already mastered. In total,
he repeated the rather unexciting comma exercise three times over. His sense of achievement at the successful process of troubleshooting the problem with the program seemed to make up for the limited cognitive challenge in the content of the exercise. His sense of achievement was palpable, and his insistence that I should also listen to the positive feedback suggests that he also enjoyed the display of his mastery over the program.

**Playing with the Rules – Cheats and Hacks**

In another interesting power struggle with the software, Brenda called me over to show how she had successfully bypassed the simulated discourse of the content of an exercise and had still achieved full marks. For reasons of brevity, the full transcription is not provided.

Brenda tugged my sleeve as I was walking around the classroom. When she had my attention, she pointed out that she had full marks for one exercise (46 out of 46). I congratulated her and she shook with laughter. The two girls working next to her quickly explained to me that she had found a way of ‘cheating’ and getting the ‘high score’ for that exercise. I asked Brenda to show me her trick, and she explained her method: ‘I click all of them, Miss, and then I just go there – Is it right? – and click, and it come all right, Miss.’

Brenda had been working on the sentence structure exercise. The exercise in question focused on sentence structure by instructing the learner to click on the ‘main idea’ in each sentence. Ideas are pretty hard things to click on at the best of times, and the exercise in question in fact required learners to demonstrate their knowledge of sentence structure by clicking on all the words of the main clause of each sentence.

When I asked Brenda to try the exercise without the ‘cheat’ I found that she had not identified clauses at all. Two sentences are provided with English translation below. Brenda had clicked on the words highlighted in bold:

- *Die meisie praat met die man met die rooi baard.* (The girl talks with the man with the red beard.)

- *Sy het die boek van voor tot agter deurgelees.* (She read the book from beginning to end.)

At first I thought that Brenda had selected words at random as her selections did not correspond to the grammatical structures of the sentences. When I discussed the exercise with her, however, her interpretation of ‘main idea’ seemed to diverge from that in the software. Although I am not sure I fully understood her explanation, she may have tried to select words which summarised the content of the entire sentence. She was certainly not aware of sentence structure in the way required by the software.

The exercise was poorly designed, since, as Brenda discovered, it could be ‘hacked’ by clicking indiscriminately on all the words in the sentence. The designer had probably intended that learners should think about the meaning of the sentences, identify the ‘main idea’ and then carefully select the words of the main clauses (which, according to grammatical orthodoxy, express the
‘main idea’ of a sentence). By developing this ingenious cheat, Brenda was not interacting with the words and their meanings at all.

Instead, she was reading a different text, which to her carried a far higher modality – this was the software scoring system with its economy of ticks and crosses, positive and negative feedback and final judgement in the form of a summary grade and printout. The game Brenda had designed involved playing with the ‘non-text’ economy of the software, hacking it to achieve a high score and displaying her mastery to elicit the admiration and attention of her peers (and possibly myself).

**Reading Interface and Interactions**

Multimodal discourse analysis should try to account for software interfaces as software rather than only as an ensemble of image, text, sound, animation and other modes. If coded structures are considered, the interface of the literacy package used at Mountainview Primary testifies to the political economy and global power relations within which the software was produced. Here I analyse the interface of the ‘main idea’ exercise attempted by Brenda, as discussed above.

Like other drill-and-practice software of this kind, teachers can select lessons suitable for each learner from a ‘bank’ of questions. Teachers cannot, however, adapt the basic coded structures of the exercises. The limits of customisation were set by the original developers who own the source code and who decided what approach to literacy teaching would be most profitable given their target market in the United Kingdom.

The software consists of a series of traditional grammar exercises, presented by an animated manga-style cartoon character positioned on the left side of the screen with buttons and exercises on the right. This interface is a visual invitation to discourse with the program. The cartoon character looks directly at the viewer, and the direct gaze echoes the imperative language of the question, creating a visual ‘demand’ (Kress & Van Leeuwen, 1996: 122). The visual address of the cartoon is extended into a metaphor of a dialogue between the user and the cartoon character through the interactional structure of the program, and the audio dialogue, which echoes text messages displayed on the screen with spoken instructions and feedback.

In terms of classroom discourse, this screen represents the teacher’s initiation of an interaction and it also conveys a visual representation of permissible discourse responses by children.

The South African localisers first chose which local languages would be most marketable. They also selected exercises to be included in the Afrikaans package. The teacher, Mr Jacobs, chose a selection of exercises for each grade (rather than the more laborious process of setting up an individual lesson for each child, which is the option advertised in the product’s marketing material). Mr Jacobs also selected a pale-skinned boy as the animated character from a paradigm of other available characters. While completing the exercise, Brenda clicked on a selection of words in a series of unrelated sentences. The words she selected were highlighted in red on the screen.

The cursor is Brenda’s avatar on the screen – it is the visual and interactive equivalent of the second person pronoun in a simultaneous narrative. Thus, the
screen represents Brenda’s mediated actions to her: ‘You have moved the cursor left. You have clicked the button’. The cursor can only be used to input text or click a word or button, and so the user’s possible responses to each question are limited to those which can be processed automatically.

Key elements of the ‘click on a word’ exercise are not apparent from the visual interface. Invisible buttons on the screen provide available responses which will be evaluated in the IRF discourse structure. In this case, each word in the exercise is a hidden button. When the user clicks on one this adds or removes a highlight. Also hidden is the list of target words which constitute the correct answer – the ‘main idea’ of each sentence. The coded logic of the exercise is not apparent from the visual interface. Neither is the logical error by which the programmers allowed Brenda to achieve a perfect score by clicking on all the words. Both of these can only be inferred as a result of interaction.

Thus, Brenda’s selections influence a relatively small part of the mediated experience. Her ‘experience’ is assembled by the logic of the program from a complex history of production decisions. A hierarchy of users are granted varying degrees of power in the production experience – from the UK designers who own the program’s source code, to the South African localisers who provided the translations, and Brenda’s teacher, who chose exercises for the class to complete and configured the installation. Compared to some of the other producers in the chain, Brenda had very little power over the representation, in accordance with her position in the hierarchy of institutionalised classroom discourse.

How Does Literacy Software Encode Classroom Discourse Patterns?

The text instructions and the question texts for all the other questions are selected by the software from a paradigm, or a set of records in a database. The individual questions and even the interface labels and images are all generated from variables – this design is a preferred development method for educational software. ‘Dynamic’ content such as this allows ease of translation, localisation and substitution of similarly structured content.

The database which stores the questions, answers and responses is also a representation of triadic classroom discourse structure and its assumptions about knowledge. According to the data structure, every question must have one of two structures: ‘fill in the blank’ or ‘click on a word’. Each question has a correct answer, which can be translated into numerical form. It also has a numbered level of difficulty, and is communicated in only one language. Feedback can be positive or negative, or can praise the user for completing the exercise correctly the first time around. Questions themselves are always in text form, although feedback and instructions can be presented in audio form as well. No questions may involve images or sounds.

The visible user interface is generated by a sequence of instructions (also known as algorithms). This code is activated when the user presses a button labelled ‘Is it right?’. The variables allow for automated cohesion with whichever future values the user might select (for example, the number of questions)
Simulated Classroom Discourse in Educational Software

or supply (textual input in a multiple choice question). Below is a representation of this function logic in pseudo-code:

Set <userAnswer> = <userInput>
If <userAnswer> is equal to <correctAnswer> for current <questionID>
Then
set <completed> as TRUE
increment <questionScore>
display <questionScore> / <questionTotal>
If <firstTime> is TRUE
Display <firstTimeFeedback> in <languageID> for current <questionID>.
Otherwise
display <positiveFeedback> in <languageID> for current <questionID>.
Otherwise,
display <questionScore> / <questionTotal>
display <negativeFeedback> in <languageID> for current <questionID>.

By encoding algorithms and data structures, source code specifies unbreakable rules of syntagmatic combination (through algorithms) and paradigmatic selection (through data structures), as explained by Manovich (2001). Computer software can thus be seen as a procedural genre. Kress explains the interpersonal relations set up by conventional procedural genres as follows:

Procedural genres, like all genres, project a world with a larger order, a coherence: whether, as in the recipe for duck à l’orange, a necessary sequence of steps to achieve the perfect dish, or in the procedure through which a scientific experiment, an industrial process, or a social event is set out. (Kress, 2003: 102)

The larger order or coherence projected by the literacy software as a procedural genre is the classroom discourse structure of the IRF. This structure is solidified and encoded in data structures and enacted through algorithms to which teachers delegate their authority. According to this order, learning can be measured in a numerical score and learning as well as literacy consists of being able to produce an answer which matches the answer defined as ‘correct’.

Cheating literacy

Reliance on educational software has been controversial among literacy specialists, who have labelled the ‘drill-and-practice’ genres an inferior model of literacy teaching, which serves to develop only the ‘meagre literacy of subordinate classes’ (Ohmann in LeBlanc, 1994: 31).

From this perspective, class divides are reinforced by different approaches to digital literacy (LeBlanc, 1994: 30). There is little awareness of this kind of critique within the Western Cape, where the new computer laboratories in schools have held out the promise of ‘curriculum delivery’ to the majority. And indeed, the schools using literacy software can point to the evidence of improvement
of children’s scores in standardised literacy tests. An internal evaluation found that schools where literacy software was used for a year were able to make substantial improvements to their literacy scores on standardised tests. Mountaintside Primary School was exceptional in showing 100% increases on Western Cape Education Department tests (Du Toit, 2005).

The general optimism is understandable, but the evidence in this study suggests that we should look beyond standardised testing to investigate more carefully what the measured improvements in literacy scores may be concealing. While improved test scores represent a significant achievement, they are not necessarily providing children with the literacy practices which are valued in society that will help them to learn. It is also unlikely that standardised tests measure the ‘new literacies’—abilities such as finding, evaluating and synthesising information, integrating image and text into a multimodal ensemble, the ability to engage in a mediated conversation over a communicational network, or the ability to program interactions (Snyder, 2001).

Although literacy exercises were used with dedication by both teachers and children of Mountaintside Primary, the drill-and-practice regimen did not seem to focus children’s attention on the communicative possibilities of language (in this case, Afrikaans). Nor did it help them develop as active language learners. Rather, these exercises negate the understandings about language learning developed over the past century: that language is learned in context and by interacting with conversational partners who can ‘scaffold’ children’s development by listening to them, helping them to adjust their utterances and responding to their ideas (Bruner, 1978; Vygotsky, 1986). In this sense, then, the imported literacy software is a ‘cheat’, which claims to deliver literacy while automating and short-circuiting interpersonal communication.

How Do Children Create Meaning from Software Interfaces?

Of further concern is the metaphor of computers ‘delivering’ education. In the interviews conducted for this study, education department officials, teachers and computer training facilitators all spoke of the need to achieve ‘curriculum delivery’ via computers. Educational media and technologies are provided to compensate for relatively unskilled teachers and for large classes. This metaphor conceals complex semiotic processes, since it assumes that computers function as a straightforward conduit for information or education, which can be packaged and provided to children. The metaphor urgently needs to be replaced with a notion of software as representational mode.

This study found that computers and software are not simply ‘delivering’ information to children. Instead, the software is always interpreted in a specific local context. The children are engaged in an active process of sign-making, and their interests often diverge from those of their educators and the creators of the software. According to the social semiotic view, meanings do not reside on the page or the computer screen, but are generated as part of a social process. Consequently, although grammars and conventional systems of meaning do exist, ‘[i]nnovation is the normal condition of all human meaning-making’. (Kress et al., 2001: 8). Children employ innovative processes of social meaningmaking
which help them make sense of decontextualised drill-and-skill exercises originally designed for children on the other side of the globe.

The uninspiring ‘drill-and-practice’ software makes few concessions to the children’s context and does without the reciprocal negotiation of meaning which is fundamental to interpersonal communication. Nonetheless, this study suggests that children use the rule-governed logic of the software as a representational resource and that this is associated with distinctive literacy practices.

The children observed in this study interpreted the multimodal software text and came to grips with its procedural nature through an investigative process and delight in some creative play with the rules, or ‘cheating’. They were seen to exercise troubleshooting and hypothesis-formation strategies which helped them to piece together the often incoherent software texts. Their goals in this process also related to the display of mastery and the scoring function of the software.

**Conclusion**

The activities analysed in the study involve children interacting with two cohesive structures—a clickable text exercise and an editable text exercise. The simulated classroom discourse of the package did not visibly engage the children. Stevie had already mastered the punctuation lesson, but was stymied when its automated cohesive relations broke down into ‘non-text’ as the result of an unpredicted typing error. He was reluctantly drawn into the activity of troubleshooting the software, and eventually succeeding in ‘debugging’ it with help from Linda. Both Stevie and Brenda were motivated by the game-like modality system of the exercises. This economy of ‘scores’ reinforced a test-oriented assessment system, a system of meanings which values the quantifiable outcome of the exercises over any process of developing a shared meaning through communication. The software was used to display mastery before me, the visiting researcher and I no doubt provided an appreciative audience. Brenda had no prior understanding of the concepts relating to sentence structure, and so she was not engaged by the content of the exercise. Instead, she took great pleasure in ‘hacking’ the software to achieve a perfect score for the exercise despite her lack of comprehension of the ‘content’ of the exercise. Further research could evaluate what aspects of literacy are being measured by standardised tests and weigh these against the opportunity costs of a drill-and-practice approach.

If we look more closely at how children used the software, we see semiotic interactions which are an extension of the power relations of classroom discourse and the global political economy of software production. We also see children resisting this power and developing skills in interacting with, manipulating and ‘cheating’ rule-governed texts. This study represents only a small snapshot of such an activity, but it is nonetheless clear that long-term investments in teacher training and further resourcing will be required before school literacy activities with computers can shift into a different paradigm—one where children use software to develop other semiotic powers, in addition to those needed to ‘cheat’ the quiz. Such a paradigm would emphasise how computers can be used to mediate communication, and would develop a procedural literacy, where users know that they have the power to write their own rules.
Where literacy is stripped of meaning and becomes a numbers game, children like Stevie and Brenda cannot be blamed for aiming for the high score. Drill-and-practice approaches to literacy teaching are a popular approach to literacy and look to be influential in the Western Cape for the foreseeable future. It is sad therefore, that along with software licences, the Western Cape Education Department is importing a limited and limiting notion of literacy and classroom discourse.

Correspondence

Any correspondence should be directed to Mr Marion Walton, Multimedia Education Group, University of Cape Town, Cape Town, South Africa (mwalton@its.uct.ac.za).

References


