First steps in creative computational thinking with natural language programming and Lego MINDSTORMS

Geoffrey Falk
Birkbeck Knowledge Lab
Department of Computer Science and Information Systems
Birkbeck, University of London
London, WC1E 7HX, UK

Keith Leonard Mannock
Birkbeck Knowledge Lab
Department of Computer Science and Information Systems
Birkbeck, University of London
London, WC1E 7HX, UK
Email: keith@dcs.bbk.ac.uk (Contact Author)

Abstract—We describe the design and implementation of a prototype Android application to help promote the development of creative computational thinking in the UK Key Stage 1 and 2 classroom. The system is based on the combination of a natural language interface and Lego MINDSTORMS robots [1], focussing on engaging children and teachers who would not naturally be interested in computer programming. The prototype was tested with groups of children at local schools. We describe the stages of development for the application and test results. The paper concludes with a brief comparison of other programming applications, and suggests ways forward for the further development of this work.

I. INTRODUCTION

The aim of this work was to develop an application that can aid the introduction of creative computational thinking into the UK Key Stage 1 and 2 classroom (KS1 and KS2). A new Computing programme of study was introduced into the UK National Curriculum from September 2014, in broad recognition that computational skill is now a basic skill that is required by everyone. It has meant that, for the first time, primary school teachers (5-9 years old) will be required to plan and deliver lessons that teach basic programming skills [2].

There are a growing number of products on the market which can be used to help teach such skills, including Bee-bots, floor turtles that follow in the tradition of Logo [3], and a range of tablet applications. However, for the beginner the overriding bias is placed on developing skills rather than inspiring creativity (see Section [V], and this may not be enough to engage the imaginations of teachers and pupils who are less enthused by the process of programming itself than by the creative things that can be done with it. This application aims to help redress that balance.

In the next section we outline the basic approach and objectives of our work. In Section [III] we describe how we determined the requirements for the work. In the following section (Section [IV]), we outline the basic design and implementation of the prototype system, including the integration of the Lego robot and Android application. In Section [V] we review and analyse the results of our experiments with the prototype application. Then, in Section [VI] we provide a brief critical comparison with two other products targeting this age group. Finally, in Section [VII] we provide some conclusions on the work and thoughts on future work.

II. APPROACH

Our system is inspired by the work of Seymour Papert and others at MIT Media Lab who since the 1980s have repeatedly shown how computational thinking can be encouraged in children. A recurring approach has been to connect abstract programming concepts with the physical world, via Logo turtles in the 1980s, physical micro-worlds, and latterly Lego MINDSTORMS robots [5], [6], [7]. As a consequence, many schools and after-school clubs have started running robotics courses, and there are competitive leagues to motivate children to become better programmers (e.g., ROBOCUP JUNIOR [8]). However, these advances are taking place against the background of declining numbers of children studying STEM subjects at higher levels [9].

There is a concern, however, that taking a isolated skills-based approach to teaching computing, even if it uses robotics, may not have the desired effect for some students [10]. Rusk et al., proposed that, for many learners, instead of focussing on design challenges, like building cars or moving robots through obstacle courses, structuring workshops around shared themes and storytelling would be more effective at engaging a wider diversity of students from early childhood onwards. Their work with the programming tool SCRATCH [12] has shown this to be true at least for older learners. Scratch allows users to create stories, art and games via a block programming interface, and now hosts several million projects [13].

Scratch has been a major success, however according to statistics from MITs site, and shown in Figure [1] the vast
majority of users are of middle school age and above, peaking at age thirteen \[14\]. No programming environment has yet appeared for younger children that can inspire the creativity and dedication of Scratch (even considering the popularity of MindCraft \[11\]).

![Age Distribution of New Scratchers](image)

**Fig. 1:** Age distribution of Scratch users

Young pupils have been found to engage easily with robots, possibly because they are comfortable with playing with Lego bricks, and doing so reminds them of toys rather than assignments \[19\]. They are also very powerful, with complex behaviours made available through coordinated use of the motors and sensors. In general, young children feel very comfortable using mobile handsets, with a huge variety of games and educational applications available on Google Play (and the Apple App Store), for four year olds and upwards. The technology for communication between Android devices and the Lego MINDSTORMS robot via Bluetooth is well established, and has been integrated into a diverse set of products and platforms.

There were four main objectives for this work:

1) **Promote computational thinking** — targeted at the current UK KS1 and KS2 computing curriculum.

2) **Possess a fast learning curve** — the application should provide an interface that young students (and teachers) can pick up and use instantly, without the need for a skills-based tutoring system.

3) **Appeal to the dramatists** — \[20\] discusses how young children start out by being “both little storytellers and little engineers”, but a division happens later on in schooling as a some children lose confidence in new abstractions that they cannot tie to concrete experience; the aim was to engage both the dramatists and the, so-called, patterners.

4) **To provide an extensible architecture**, which can easily be adapted to a changing, Internet-of-Things world.

The approach has remained closely aligned to using storytelling as a method for engaging children in programming by integrating an Android based application with the Lego MINDSTORMS product, utilising a novel, natural language style interface.

III. Requirements Gathering

Three methods were used to gather requirements for the project — Interviews, Document Analysis, and Prototype development and testing; in this paper we will focus on the interviews with the children.

Thirty children were interviewed and observed over a two week period using various pieces of software for both developing general programming skills and for controlling a MINDSTORMS robot, which offer basic remote control over the robots motors. We now consider three of those children, who had typical responses from the group.

Bruce (aged 9) performs near the top in his class in English and Maths, and is quick to master computer interfaces. He has been given a Wordpress site, which he sporadically updates it with performances and puppet shows. He plays the piano and shows aptitude in all subjects, but he is especially keen on creating music and writing stories. He could be classified as a “dramatist”.

He quickly lost interest in the various remote controller applications available for MINDSTORMS. He showed interest in writing stories for the robot to act out, but when he was shown the available tools he wasn’t sure how to use them, and what they did. When told that they were to help him tell a story with the robot, he couldnt say how he would achieve this. When asked about how he would like to control the robot, he mentioned that he’d like to create an adventure story with the robot.

Leo (6) is beginning to write, but struggles in school with Maths in particular. His interests are street dancing, art and football. He is not confident using the computer, and generally plays games on CBBC and CBeebies, and getting print outs which he then colours in.

He enjoyed using the basic applications, and was skilled at moving the robot around the room. When asked what he would like to get the robot to do, he mentioned “chasing people” and “you tell it do something and it does it”.

Lilly (5) got on really well with the basic MINDSTORMS applications. Although her reading level wasn’t at a level where she was confident to read the words in the “basic” interface. When it was explained what she would have to do, she said she would like to tell a story about a fairy robot who “does magic”, but couldn’t explain what she would have to do to achieve this with the interfaces of the standard applications.

Requirements gathered from this stage therefore reinforced the research that storytelling could be an good way to engage the “dramatists”, and that it was important for the application to allow for complex interactions making use of both motors and sensors. Enough details had now been gathered to define the final requirements and the following subsections discuss these.

A. User Requirements

Users should be able to write and run programs using complete English sentences via an NLP interface. The interface should be straightforward enough for a literate 6 year old to use, and allow users to do the following:
write a program, or “story”, in full English sentences on an Android device
adapt a “story” written by another user
save their “story” for later use
run and test their “story” on a connected MINDSTORMS robot

B. Technical requirements

The main technical requirements were:
to implement an Android application that remotely controls an MINDSTORMS robot
to coordinate the robots actions with Android sounds and music
to map this coordination to the sequence of instructions dened by the user
to translate the instructions from NLP sentences as dened by dened RLIT ontology (see later)
to upload sounds to the MINDSTORMS robot
to read information from the following robot sensors: ultrasonic, sound, light, switch
to read the “tacho count” (motor rotations) from each robot motor port
to direct the robot to move in a direction, rotate, play sounds and beep
to work on both handset and tablet size screens.

C. NLP specication — RLIT

Part of the specification stage was to deine in detail the ontological rules governing the NLP language, RLIT.

• The language needed to provide a range of sentence constructions that would allow the user to control a MINDSTORMS robot via Bluetooth, including setting motor actions, reading sensors, and playing sounds and notes.

• The language would also need to allow users to coordinate with this music tracks and spoken dialogue from the Android device.

• The language would not need to provide users with the fine grained control over the motors and sensors that, for example, the Lego software does. It is sufficient to have a sentence like “ROBOT moves forward a little and steadily”.

• The language should allow for extensions to allow for controlling variables such as speed, distance, degrees of turn, and sensor thresholds.

• The language would need to include the ability for looping statements, and event handling. This was essential to address the programming objectives in the Key Stage 1 and 2 Computing curriculum.

• The language would need to support concurrency, so that more than one event could happen at the same time, for example moving the robot around whilst playing music from the Android device. This would bring a freedom to the users programs and also allow for interesting situations that would require the user to debug their “programs”. Again, this is a stated objective in the Key Stage 2 curriculum.

IV. DESIGN AND IMPLEMENTATION

RL is designed as a multi-tier application, based on Androids MVC (Model View Controller) architecture, using the Kotlin programming language and its coroutines functionality for the concurrency aspects (Figure 2). Kotlin was preferred over Java as it has a more concise syntax, with higher functionality, and is a preferred language for the Android platform.

A. UI and UI Controller Layers

The UI layer is composed solely of XML and resource files which are accessed and inflated by the Activity and Fragment classes in the UI Controller layer. The controller layer also provides access to the RLIT model layer, which contains classes implementing the ontology of the RLIT natural programming language. The controller layer connects with a storage data layer that is composed of an SQL database accessed via the EXPOSED API [15], allowing users to save and load stories and sounds.

After the controller layer connects via Bluetooth with the robot device, all subsequent communication with the robot is managed by a background service, which the controller layer binds to. The controller sends, via Messenger objects, all the information required for the service to communicate with the robot, including configuration choices and RLIT program instructions. Robot and other program events are then relayed back to the UI controller layer in a similar manner.

This separation of concerns provides maximum flexibility for expansion and adaptation in the future. By separating out the UI layer from the controller layer, any number of different permutations of design can be used without affecting the rest of the application. Similarly, the RLIT model can be used independently in a different application environment.

B. RLIT Layer

Like other NLPs, RLITs foundation unit is the “sentence”. Each sentence represents a distinct instruction for controlling either the robot or Android device. The phrases within each sentence modulate how and when these instructions are performed. In turn, a set of sentences make up a “story”. The “story” defines how the sentences are executed relative to each
other. Once the story is written, an interpreter analyses it and converts it into a “program” that can be subsequently executed. Figure 3 illustrates the mapping between RLIT components and Robot program components. Unlike a fully developed NLP, the user is guided through writing an RLIT sentence, step by step. This ensures that the process is as simple as possible, especially for Key Stage 1 pupils.

Phrases are organised into hierarchical structures defining their exact ordering in sentences. Figure 4 shows a fragment of this structure.

Two possible routes for RLIT phrase combinations are shown in the figure; one for combinations that instruct the robot to move, and the other to instruct the robot to wait for a target, reading from the ultrasonic sensor. Each group of phrases has an ID, and their “children” are represented by a directed graph. For example, if the phrases First then ROBOT are selected, all phrases that have the parent ID of ROBOT (ID 10), are displayed to the user. If the phrase moves forward is selected, then all phrases with parent ID 105 are shown. This process continues until all possible phrases are exhausted.

RLIT was developed according to an ontological framework which provides sufficient power for users to program the robot in ways that can only otherwise be achieved using Legos native API. However, as indicated earlier, fine-grained control was deliberately omitted to provide a soft storytelling environment for users rather than a mathematically exact one. However, this functionality can be added at a later date (see Section VII).

The role of the interpreter in ROBOliterate is to take the stories constructed in RLIT and convert them to sets of instructions that Android device and robot can execute. The interpreter ascertains which instruction type to create based on the InstructionID provided by the verb phrase in each sentence. Each instruction is represented by a unique ID, and several RLIT verb phrases can share the same instruction ID if they describe variations on the same instructions.

For example, the RLIT verb phrases move back, move forward, and stop moving all share the instruction ID INSTRUCTION_ROBOT_MOVE. The verb phrases turn left and turn right share the instruction ID INSTRUCTION_ROBOT_TURN. On the other hand, there’s only one way to beep, so the verb phrase BEEP is unique in being the only verb phrase to have the ID INSTRUCTION_ROBOT_BEEP.

Once the instruction type is extracted, it is instantiated by the interpreter with parameters that are dened by an analysis of the eld values of all of the phrases in the sentence. Any particular phrase eld value means something unique depending on the instruction to which it is attached. Once the instructions are built, they are saved as a “Program” — a simple list of instructions that in turn contain all the data necessary for the robot control classes to use during program execution.

The design had to allow for the later substitution of new models of MINDSTORMS robots in place of the current EV3 device. In addition, the design needed to allow for future versions where multiple devices can be controlled, or where multiple Android devices could work together. For this reason, Androids Bound Service architecture was chosen, since it allows for multiple applications to connect to the same service [7]. Using this architecture also allows for the loose coupling required to allow new robot models be substituted without affecting the other components of the architecture.

Five different families of robot commands are provided: 1) direct commands that control movement and sound, 2) system commands to upload files, 3) “reply” commands that obtain readings from the motor ports, 4) commands that obtain readings from passive sensors (e.g., switch) and active sensors (e.g., sound), and 5) specialist commands that send and receive information from digital sensors (e.g., the ultrasonic sensor).

The software was implemented and tested in four stages:
• Phase 1: Implementation of the UI Layer and RLIT model.
• Phase 2: Implementation of the robot service layer.
• Phase 3: Integration of the UI Layer, RLIT model, and Service layer.
• Phase 4: Refinement of the UI layer and RLIT model based on feedback from user testing.

The software was tested iteratively throughout its development, starting at the prototype stage, and progressing through test case scenarios to integration testing, system testing and finally user evaluation.

V. RESULTS AND ANALYSIS

Thirty children aged between 6 and 10 were selected by the School Principals from three local schools on the basis of their range of skills. The children were also identified according to their interest in English and art.

Five thirty minute sessions were conducted, each with several students and a teacher observer. The teacher observers included the Principal of the school, an IT coordinator, and an Art teacher. There were two testers present at each session, one who ran the session and the other recording the events with video and still cameras.

The structure of the sessions were as follows:

1) The students were shown a short demonstration story with the robot. The story was about a lonely robot, who turns around and moves away when asked by Android to play. Then Android offers to play some happy music to cheer the robot up. When the music starts, the robot starts spinning around and beeping.

2) The children were then told that it was their opportunity to play with the robot and Android. They were introduced to the application ROBOLITERATE, however they were given no instructions on how to use it. Instead they were told they could ask for help if they got stuck. The children worked in pairs, using a variety of Android based tablets and handsets.

3) The children were observed using the application for ten minutes. When they were ready to test what they had written, they were guided through the steps of connecting to the robot.

4) They were then observed as the program executed, and asked questions such as “What is happening now?”, “Is it what you expected?”, “Why is this happening?”, “What went wrong?”, and “How would you change it?”

5) Each pair was then given another chance to go back to their story, make changes, and run the program once again. If time permitted, then they carried out a third cycle.

6) As a plenary, the children discussed what they liked and didn’t like about the application, and how they would like to change it.

7) Observations were also recorded from the Principal, IT coordinator, and Art teacher.

It was interesting to note that it was a younger group of children (the 8 year olds) who showed the most fearlessness and joy in using the application. Their shouts of delight, also noted by the Principal, showed that they were experiencing real excitement at getting their robot to perform.

The older children were more hesitant, maybe because they were more wary of getting things wrong, however though their approach was more tentative, by the end of the half hour some were beginning to think more about the stories they wanted to write. There was some disappointment that the robot couldn’t do more things, and the younger children couldn’t quite understand why it couldn’t do things like jump or do exactly what they told it, however this stimulated an interesting plenary discussion about what it means to program a robot.

The use of audio added a very important enriching layer to the experience, enabling the children to personalise their story. It was unfortunate that the EV3s speaker was so quiet, meaning that unless there was complete quiet, no one could hear it!

From these tests, there are promising signs that this approach could provide a method for introducing computational thinking into the classroom in an entertaining and engaging way for children most interested in performance and telling stories. Evidence of learning was found by the testers and reinforced by the teacher observers, that matches many of the stated objectives of the Computing programme of study for Key Stage 1 and 2.

It had already been found during earlier user testing that the application can be used to teach Key Stage 2 objectives, in that they were engaged in “designing, writing and debugging programs to accomplish specific goals with physical systems” (KS2 objective)

However, the critical test was to see if computational thinking could happen in a classroom environment where children are not being led “recipe-style” through scripted learning points, but through exploration based on their own imaginations and creativity.
Initial results based on this user evaluation show that this applications approach is a method that can be used to facilitate this, serving as a basic introduction to creative computational thinking. There was no explicit goal given to the evaluation groups at the beginning of the session; after they were shown an example program, they were simply encouraged to have fun and explore the application themselves. It was emphasised to the children that there was going to be no right or wrong answer. By the end of the thirty minutes, many children were already engaged in aspects of computational thinking and self-setting goals for the next story they wanted to write.

Some of the observed learning included:

- learning that programs execute by following precise and unambiguous instructions,
- practice creating and debugging their own programs,
- using logical reasoning to predict the behaviour of their programs,
- investigating why their program was not behaving as expected.

These are stated objectives of the new Key Stage 1 curriculum.

VI. CRITICAL COMPARISON

Having conducted the user evaluation, we wanted to compare these findings with results from testing other applications that teach computational thinking to Key Stage 1 and 2 children. In particular we wanted to focus on engagement in learning, and whether ROBOLITERATE can stimulate the dramatists to learn more than other applications can. Given the lack of space we will only discuss the reaction of one group of children, although we examined the whole cohort. We consider “Engagement” and the extent to which the application promotes “Computational Thinking”.

We chose two of the most visible applications for the iPad — “Hopscotch” [16] and “Move the Turtle” [4], which are both aimed at children aged 8-11 years old. Hopscotch is heavily inspired by MIT’s Scratch, allowing children to build program routines that control cartoon characters by dragging blocks onto the screen. “Move the Turtle” is similarly aimed at 8-11 year olds, and draws its inspiration from Logo, enabling children to program a turtle to move and draw patterns on the screen. Screenshots of these applications are shown in Figure 6.

![Fig. 6: Two current iPad applications for teaching programming](image)

A. Engagement

Bruce would probably classify himself as a dramatist, as noted before, in that he loves music and acting. However, he is also very quick on the computer and iPad, and was eager to test the programming apps we downloaded for him.

Both Hopscotch and “Move the Turtle” engaged Bruce’s interest for a while, and he returned to them a few times over a two week period. It appeared that both applications had different incentives for him to return, however neither were strong enough for him to want to keep on returning to for long compared to, say, Garage Band [17], which he was constantly revisiting, making and tweaking his musical compositions.

“Move the Turtle” has a very strong tutoring system which leads children step by step from programming the turtle to move, to navigating a maze, to drawing fractals. Bruce enjoyed the reward system and was proud that he had already finished the first eight mini-levels and had reached Level 2. It appeared that achieving a score was the main motivating factor for this application, and he wasn’t so interested in the movements of the turtle, because having got to Level 2, he didn’t continue.

“Hopscotch” was more interesting for him — after a week, he had managed to move three ‘characters’ across the screen, painting wavy trails behind them, and was keen to show us what he had achieved. By the next week, however, he had not picked up the application again. With “Hopscotch”, he was pleased with the results of his artwork achieved through programming, but there was a ceiling of interest that was reached quite quickly before he moved on. The issue with “Move the Turtle” was that the rewards were insufficient to hold his interest. This is an issue with many applications on the market, which focus on narrow skills-based programming tasks but do not do enough to stimulate the minds of users and empower them to use programming for their own purposes.

Bruce found that ROBOLITERATE engaged his interest for longer. It gives much more free reign to be creative, to program movements, sounds and dialogues, and to do so in a tangible physical space that gives the feel of a real performance. Clearly, more work will have to be done to see whether an approach like ROBOLITERATE can, in general, engage creative minds longer than the other applications, but the early indications are positive.

B. Promoting computational thinking

In additional to testing engagement, the applications were analysed together with ROBOLITERATE to ascertain the coverage of the key concepts and practices for computational thinking [18]. Table 1 summarises the concepts covered by each application, and displays an even coverage between the applications:

It is hard for an application alone to encourage computational thinking practices. These practices are: 1) being incremental and iterative, 2) testing and debugging, 3) reusing and remixing, and 4) abstracting and modularising. It is the environment that is built around a tool, and the context within which it is used, that are at least as important in influencing the development of these practices. One of Scratch’s great strengths is its online community, that encourages the reusing and remixing of other members projects.
Further work has been identified which will improve ROBO-LITERATE’s stability, educational aspects, user-friendliness, and enhance the rewards for the dramatists. Crucial improvements include the following:

- Re-introducing “procedures”, but designed to t with the RLIT interface. This will allow for complex behaviours to be used right from the beginning, and later adapted and refined by users.
- Extend the language and interface to allow for control over variables and data.
- Add adaptive feedback layers, that not only help users with the interface, but also give guidance on how the explore the full range of actions available.

The approach gained lots of interest from students and teachers alike, with one Principal sufficiently interested to want to start investing in MINDSTORMS for his school.

**REFERENCES**

[13] [https://www.codeclub.org.uk/projects](https://www.codeclub.org.uk/projects)
[17] [https://www.apple.com/uk/mac/garageband/](https://www.apple.com/uk/mac/garageband/)

**TABLE I:** Coverage of computational thinking concepts

<table>
<thead>
<tr>
<th>Concepts</th>
<th>Hopscotch</th>
<th>Move the Turtle</th>
<th>RoboLiterate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequences</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Loops</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Events</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Parallelism</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Conditionals</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Operators</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Data</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>

The products in this study don’t approach Scratch in terms of being able to encourage these practices, however they all go some way to achieving it. A great strength of ROBO-LITERATE is how its design encourages debugging, which was witnessed during the user evaluation. It explicitly shows and updates the status of a running program in its console window, similar to a debugging window in an IDE. Many children used this to discuss what was happening with their program and why it wasn’t behaving as expected.

**VII. CONCLUSIONS AND FUTURE WORK**

ROBO-LITERATE introduces a different way to approach programming and computational thinking for Key Stage 1 and 2 children. It has the potential to open up programming to children and teachers who feel more comfortable with words and stories than they do with patterns and symbols. It could provide a means for children to explore theme-based programming that is cross-curricular and not tied to a specific skills-led agenda. At the same time it has been shown in testing to be rigorous enough that, in a short thirty minute session, many aspects of computational thinking are touched upon, covering several aspects of the English Computing curriculum.

In general, ROBO-LITERATE has met our original objectives:

- **Promoting computational thinking:** ROBO-LITERATE gives students experience in sequencing, looping, events and conditions, and introduces them to the practice of creating and debugging simple programs, and sharing work with others.
- **Possess a fast learning curve:** in testing, all children could master the interface quickly, and some of these children had English as a second language. However, more work will need to be done to introduce children to the range and depth of what they can create, and not just how.
- **Appeal to the dramatists:** from the evaluation, there was much excitement and interest, with children already planning what stories they wanted to tell. More work will need to be done to help dramatists tell their stories, however, with further examples for them to work with, and richer functionality and behaviours for the robot.
- **Be future ready:** the aim of the system design was to ensure that the application could be adapted easily for later iterations of MINDSTORMS. Only time will tell if this has been the case.