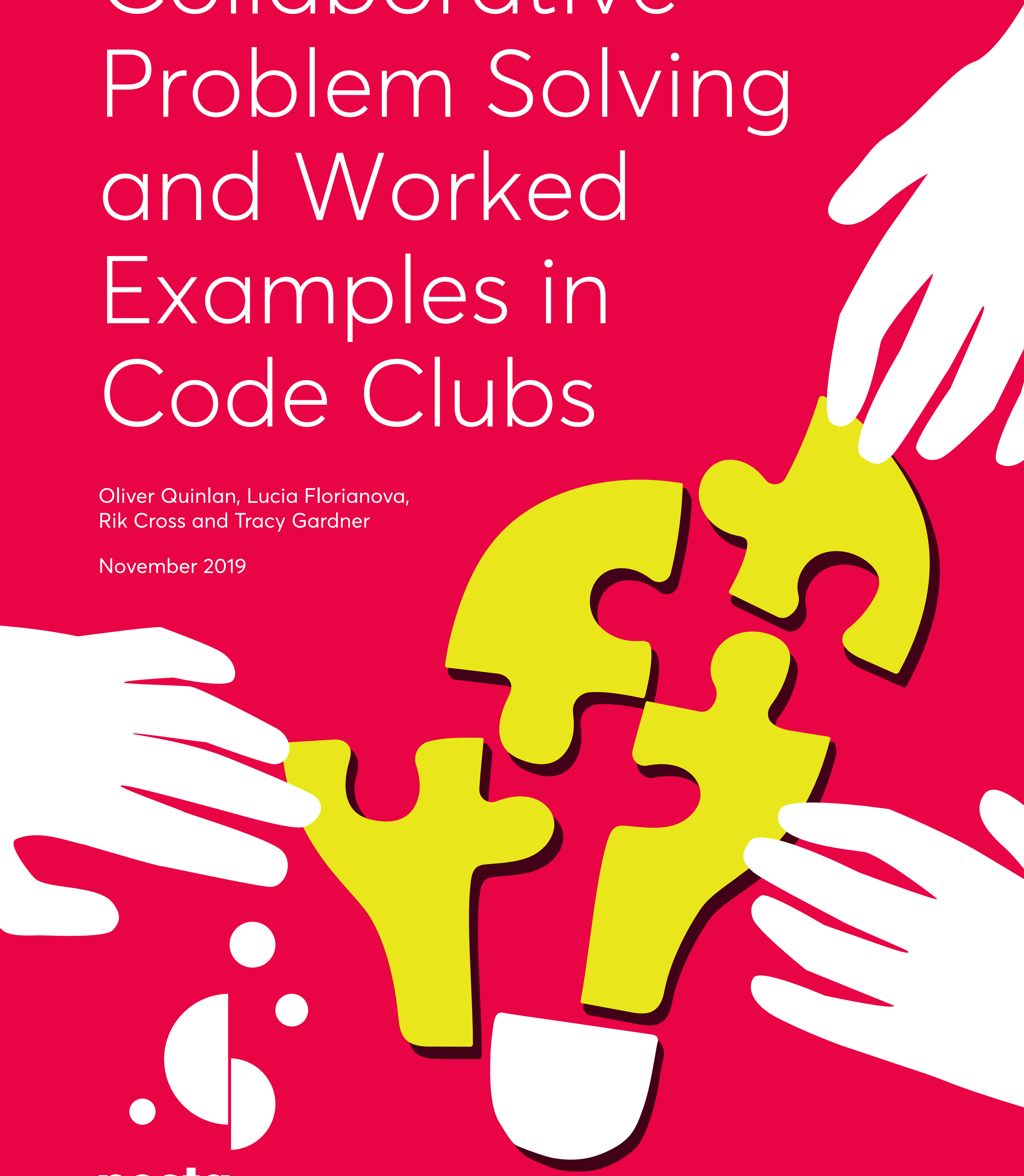
Collaborative Problem Solving and Worked Examples in Code Clubs

Oliver Quinlan, Lucia Florianova,
Rik Cross and Tracy Gardner

November 2019



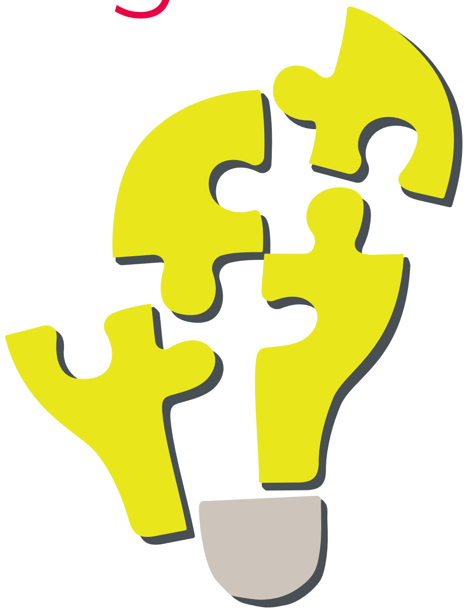
nesta

nesta.org.uk

Acknowledgements

With thanks to the Code Club leaders and young people for taking the time to try this new approach and generously give their feedback.

Collaborative Problem Solving and Worked Examples in Code Clubs



November 2019

Summary

Code Club is an international network of volunteers and educators who run free coding clubs for children and young people. A volunteer teams up with a community venue, such as a school or library, to run an after-school club, using specially created Code Club resources. Code Club is a programme of the Raspberry Pi Foundation, an educational charity focused on computing and digital making for young people.

In the Autumn term of 2017 we worked with six Code Clubs in schools in England to try a new approach to teaching programming and explore the impact it had on collaborative problem- solving and the understanding of programming concepts. We provided children with 'worked examples' of completed programmes and questions about how they worked to encourage discussion and for them to manipulate and explore (Sweller & Cooper, 2009). We compared this to our usual approach of providing step-by-step instructions for children to build projects from the start.

The children worked on these projects using the Scratch visual programming language. The resources focused on teaching them to define procedures in Scratch, through a process of creating 'custom blocks'.

We visited the schools to observe the final session in the project, and interviewed the adults leading the Code Clubs. Using Nesta's 'Taxonomy of collaborative problem-solving' as a framework, we analysed the nature of the collaborative problem-solving observed (Luckin et al., 2017). We found that the worked examples approach could have benefits for the process of understanding programming concepts, but that there were barriers to the collaborative problem-solving taking place in the clubs.

Key findings

- Fostering collaborative problem-solving takes a structured approach, and needs to be closely facilitated, particularly in informal learning environments. This might involve setting up the environment to strongly encourage collaboration, or explicitly giving children roles to take.
- Worked example-based resources can encourage and allow space for children to take an exploratory and creative approach to programming.
- Worked examples can focus children's attention on key aspects of the learning objectives, compared to the step-by-step instructions where their focus can end up on other aspects of building the project such as presentation.
- It can sometimes be difficult for adults to see the progress with worked examples since the children hadn't built them from scratch.

Next steps for the Raspberry Pi Foundation

- Put in place strategies for volunteers to facilitate collaboration such as children sharing computers and guidance for volunteers.
- Develop more learning resources with a worked example approach, particularly for more complex concepts to help children focus on mastering them.
- Adapt our worked examples approach so children make their own changes and contributions to the code that adults can use to understand the progress they have made.

Introduction

Background

The Raspberry Pi Foundation's Code Club programme supports after school programming clubs for children and young people. In over 6,000 clubs in the UK, children work with teachers and volunteers to create programming projects based on provided project guides. Children are encouraged to take a creative approach, making the projects their own and learning from each other. Clubs take place in an informal context, usually after school or during lunch times.

Code Club projects usually take a step-by-step approach. Children start from a blank page and follow instructions to create a project, with encouragement to customise or make it their own along the way. In researching alternative approaches to learning we discovered the strong evidence from mathematics education for an approach using 'worked examples' improving children's ability to solve problems (Sweller & Cooper, 2009). Rather than solving problems from scratch, the worked examples approach provides learners with examples of problems and their solutions for them to analyse. We were interested in whether this approach could be used to increase collaboration in Code Clubs, as the worked examples can provide an artefact that children can have a discussion about. We designed an approach to test that involved posing questions to the children to encourage them to explore and discuss the examples with others, tweaking the code as they worked together to understand how it had been used to solve a problem.

In this project we worked with a group of clubs, providing them with resources to teach the programming skill of defining procedures using the Scratch programming language. Clubs were randomly allocated to take either a step by step approach or a worked example approach to developing this skill. In the final week the children were all set a challenge to demonstrate their understanding of the skill, and we visited the clubs to observe learning and to interview the club leaders about the progress that had been made.

Aims and objectives

The pilot compared two different task designs to determine their effects on learning and which one better fosters collaborative problem-solving: step-by-step problem-solving or worked examples.

Research question:

Which results in more success in collaborative digital making: attempting problem-solving practically or studying worked examples of problem-solving processes?

Aims:

1. Explore whether worked examples and discussion prompts encourage more collaborative digital making.
2. Explore whether worked examples and collaborative problem-solving have an impact on children's learning in a new programming topic.

Methodology

The pilot studied the environment of six different Code Clubs for year 3 to year 8 students. Three clubs had a mix of age groups, and three clubs worked with one school year group only. All Code Clubs ran in a school environment for one hour. Most were in extra curricular times such as before or after school or during lunch times. One took place during lesson time. Clubs were selected in the London and East of England regions. A Code Club regional coordinator drew up a list of clubs that were well established with leaders who might be open to participating in a research project. They were contacted and those that responded positively became part of the project.

Schools were randomly allocated to two groups and provided with six Scratch projects designed by Code Club (one project for each week of the trial). Group A consisted of two schools and Group B consisted of four schools. Projects in both groups consisted of the same learning content, programming concepts and context, but followed different task designs.

Group A received projects providing a step-by-step guide. This was the control group, where students followed the usual Code Club style involving creating their program from scratch. Group B received Worked Example projects that took a new approach to teaching the same concepts. Students had a series of instructions encouraging them to explore how the projects worked by tweaking them and changing parts of code. The first (introductory) project, and the last (assessment) project were same for both groups.

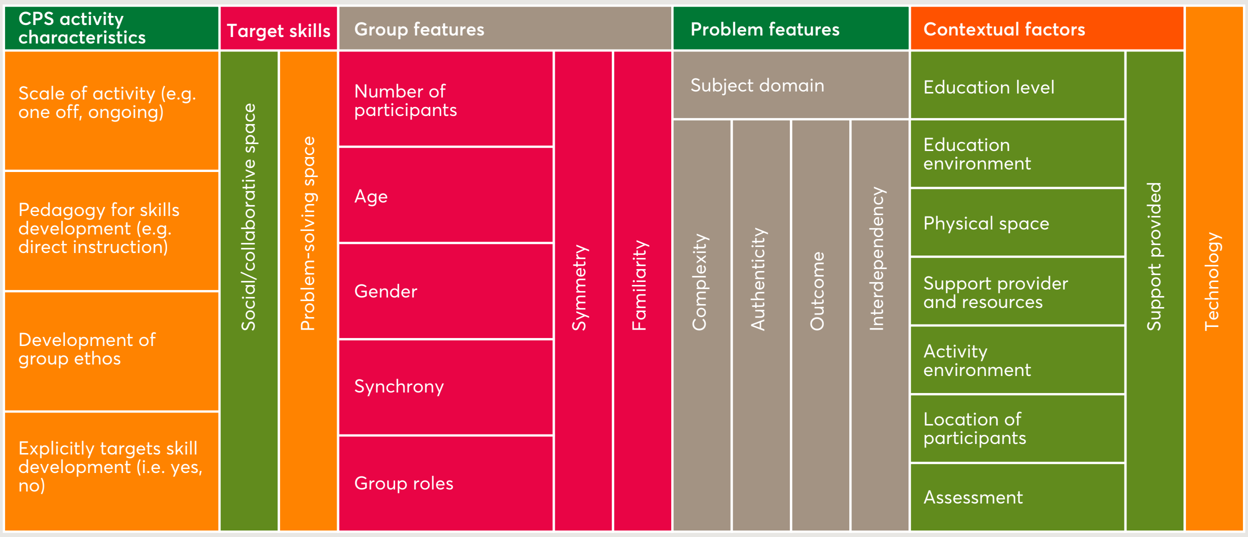
We used a combination of qualitative research tools: participant observation during the final session and a qualitative interview (30-45 minutes long) with each club leader. We also talked informally with students and examined their completed final project, to explore their understanding of their solutions.

The analysis was approached from two directions. Firstly, we explored the collaborative problem-solving that had been observed and discussed when visiting the clubs using Nesta's framework for collaborative problem-solving (Nesta, 2017: 49-51). Secondly, we looked at the children's engagement with the programming skills the resources aimed to teach.

Collaborative problem-solving

In *Solved! Making the case for collaborative problem-solving* Luckin et al., set out a taxonomy of collaborative problem-solving based on the literature into this approach to learning. This taxonomy has six non-hierarchical, interconnected domains to describe and classify collaborative problem-solving in practice. They cover all aspects of the experience of collaborative problem-solving, from the specific features of the activity and the problem, to wider features of the group and contextual factors affecting its members. We used this taxonomy as a framework for analysing the activity we observed in Code Clubs involved in this project, noting features of what was observed against each area of the taxonomy.

Taxonomy of collaborative problem-solving



Luckin et al. (2017) 'Solved! Making the case for collaborative problem-solving.' p.25. London: Nesta.

Programming concepts

The learning resources for this project consisted of a six week course using the graphical programming language Scratch. Students manipulate blocks in Scratch to create programs (or projects), fitting them together to define the actions and behaviour of objects (known as sprites) displayed on screen. Each block has a predefined function, but users can also define their own blocks by chaining existing blocks together. Such a chain performs a series of instructions that can be packaged into a new block and used in their program again.

The focus of the resources was learning how to define your own blocks, the Scratch equivalent of defining procedures in other programming languages. Packaging sets of instructions into blocks allows students to simplify the organisation of their programs, and facilitates making more complex projects. It also allows them to reuse code for efficiency, and abstract away details to assist with addressing more complex problems. It is a topic that is often not taught until students are very experienced with Scratch and is rarely the focus of early teaching, although we felt if presented clearly it could be accessible to beginners. This programming topic was thus chosen as the children in this project were likely to have no prior experience of it. The projects demonstrated the technique of creating blocks through tasks such as navigating a maze, creating an interactive quiz and instructing a character to perform dance moves.

In the final session both groups were set the same challenge; to draw a series of geometric shapes using a pencil sprite in Scratch. These shapes were initially simple (such as a square), to let students develop a basic understanding of the concept, encourage them to explore how to create a basic algorithm using loops and repetitions and define the shapes. The task then became more complex and required drawing various combinations of the initial shapes (Figure 1). The task could be completed without students creating their own blocks, but doing so would considerably simplify the problem-solving process (Figure 2)

Figure 1: Shapes that students were challenged to draw during the final task

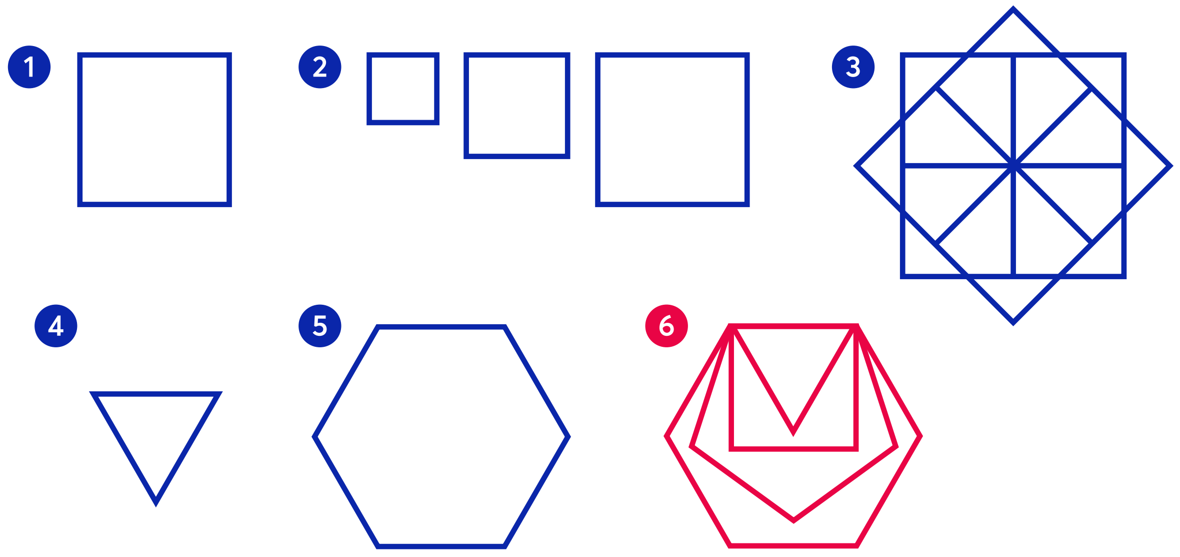
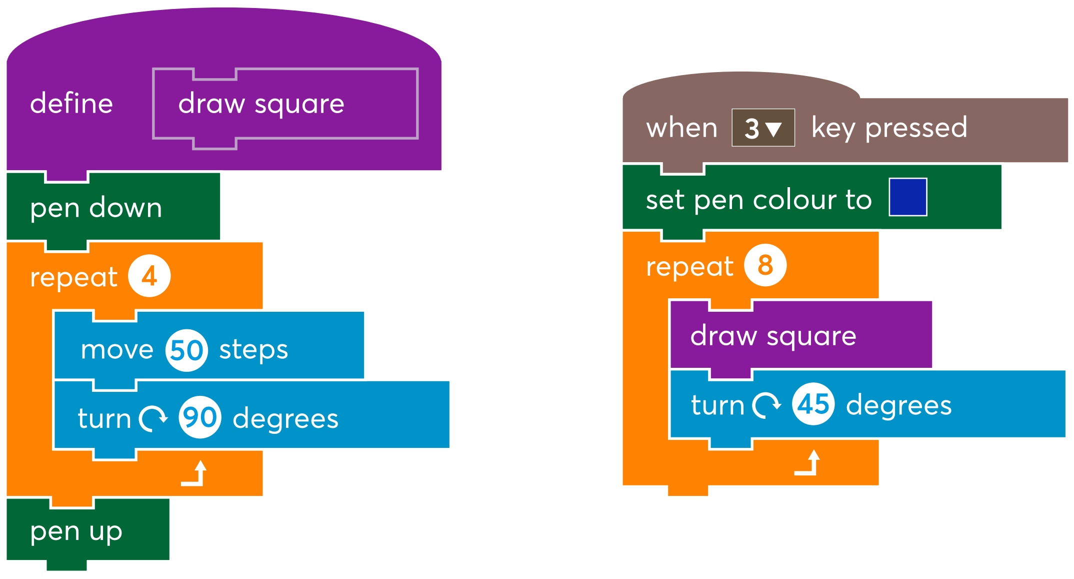


Figure 2: Defining a function to draw a square (chain of blocks on the left), which can then be used to draw other shapes, in this case the shape number 3 (chain of blocks on the left



```
define draw square
pen down
repeat 4
    move 50 steps
    turn ↻ 90 degrees
pen up
```

```
when 3 ▾ key pressed
set pen colour to
repeat 8
    draw square
    turn ↻ 45 degrees
```

Findings and results

Collaborative problem-solving

We aimed to find out whether the resources we designed based on worked examples encouraged more collaborative digital making. We were not able to observe evidence of a difference in the collaboration that took place between the two groups. However, our observations and interviews did provide insights into how collaboration works in the informal settings of Code Clubs.

Making collaborative problem-solving successful depends on the right combination of many aspects; such as educator support, task design, and group features. Achieving this combination is often challenging in practice, as we found from working with the schools in this project.

Collaboration

Although some students worked together, they were not observed collaborating in the strict sense. Interviews with adults suggested this was the case throughout the sessions. Children helped each other, but did not systematically discuss problems and join different knowledge they had. Instead, they seemed to seek others' advice and expertise when stuck; either from their friends who knew what to do, or from a teacher. They were not generally able to move on without such help when getting stuck during the final task.

Volunteers emphasised that the task we observed was much more challenging than previous tasks. Several asserted that collaboration, in terms of 'helping each other', had taken place more in previous sessions.

"They are great at actually supporting each other, like 'a' or 'b', when she was stuck, the other one showed her what to do."

"So I would have leaders in groups who were kind of doing my job. They enjoy showing and helping other kids so it's good."

Nevertheless, based on our observation of the group communication and the adults' descriptions, this peer support activity did not address the range of factors in the taxonomy that would allow it to be classified as collaborative problem-solving. It seemed that helping peers was less about lateral thinking and synthesising knowledge, and more about one person having completed a section of the task and another receiving their explanation of how they had done so.

We did observe some limited interaction that would be defined as collaboration. In one particular example, two boys compared their code and discussed how they had achieved the same result in different ways. They were comparing, contrasting and discussing, and appeared to be exploring their approach with a high level of understanding. They were identified by the teacher as particularly able children and seemed to have knowledge symmetry, ability to reflect, enthusiasm and engagement. They were also friends.

"They are quite good at talking to each other. They would just stand up and walk around and look at someone else's code to see what's working there. So I will always go around and be like: Look, this guy can do this! And this person has done this! And then you get kids go like No, no, no! And they go and compare their code to someone else's. And where possible, they would change their [code] if they remember."

These sorts of interactions were observed in several cases, but were not happening systematically across the children in the project. They suggest the importance of considering the grouping of children according to the group features in the Nesta collaborative problem-solving framework. In the example above there is some symmetry (sharing knowledge) and familiarity with working with one another evident, but this is by chance rather than design, and facilitated through children moving through the classroom. The potential for collaborative problem-solving could be maximised by forming groups based on different levels of knowledge so that this can be shared when considering the problem. The process would also benefit from building familiarity with collaborative problem-solving through modeling by the adult or other participants, or through low-challenge activities designed to practice the collaborative approaches to activities, as are promoted in the *Kagan Cooperative Learning approach* (Kagan & Kagan, 2015).

Achieving collaboration

Following Nesta's taxonomy of collaborative problem-solving, we identified a number of aspects that may have hindered successful collaborative problem-solving in this pilot and could be improved in the future. We found that collaborative problem-solving was primarily impeded by environmental factors, how technology was used and lack of guidance for adults on how to facilitate collaborative problem-solving. There were also some issues with group formation, interdependency and targeting collaborative skills development.

Environmental factors

Code Clubs generally take place in informal situations either after school or during lunch times. This informal nature means that facilitation varies across different clubs, and the sessions are often loosely structured and driven by the children themselves. Several adults expressed that Code Club was an experience that was different to lessons, and that it should stay this way. They would be reluctant to structure it too closely as they perceive children enjoy the freedom they have.

'I didn't sit them. Because it's a club, not a class. So they get to choose their little group and they work in them.'

'They come and they sit next to anyone they want; they come and get their Chromebook, get their cards, sit where they want. [...] But that's ok, because it's not a classroom; that's what I tell everybody.'

Giving the children tightly defined collaborative roles or expecting the adult to take a strong lead in mediating interactions is rarely appropriate for Code Clubs, and collaborative problem-solving has to be encouraged by more subtle approaches related to the setup of the environment and the task design.

Technology and equipment

Technology can facilitate collaboration, such as by providing a channel for communication (Nesta, 2017:51). However, in the case of sessions we observed the abundance of technology hampered the opportunities for collaborative problem-solving. In every school we visited, children had access to enough computers to have one each.

This meant adults encouraged children to use individual laptops rather than sharing, which resulted in an individualistic focus to the work. The children did not need to discuss when they had different ideas and approaches, everyone could simply follow their own ideas on their own device. We only observed them turning to others when they felt stuck, something which was affirmed by our interviews with adults.

Future projects exploring collaboration would benefit from more explicit instructions on the use of equipment. It was mentioned in guidance to adults that we wanted children to work together to produce their work, but this was not interpreted as them using the same computer in pairs or groups, which would be likely to have encouraged much more collaboration.

Adults' roles

When discussing collaboration and working together, it was apparent that adults framed the activities as tasks to be completed individually, with the chance to work together coming when a child reached a point at which they could not continue without help. This resembles the mode of working often used in traditional lessons in schools.

There may also be preconceptions about the nature of the subject. Programming can be seen as a solitary activity, especially as only one individual can realistically be manipulating a computer at a time. In more sophisticated programing, tasks could be split between programmers working on their own computers and then recombined to make a whole. In the context of Scratch, which is used for short time periods and by children with developing IT skills, this kind of collaboration would be hard to achieve.

One adult we discussed collaborative problem-solving with expressed that he would need to undertake some training in order to be able to effectively facilitate the kind of collaboration we were looking for. Another volunteer felt the same way:

"I think they'd be able to, but it would have to be really structured. You would need to assign roles [...] and we will see how we can combine all those three things. That would be the way I would approach it."

The role of the adult is clearly important, and the adults in this project would have needed much more support on what collaborative problem-solving is and the techniques to encourage it in the form we were looking for.

Group formation

The trial did not involve any deliberate formation of groups. A more explicit focus on creating groups and coalescing them around a shared goal and identity as part of the activities, or setting particular roles within groups, could have fostered more collaborative problem-solving. In the relatively short time available for a Code Club it can seem counterintuitive to spend some of it on group formation, but if collaborative problem-solving is one of the aims then an early investment of time in this would be needed.

Interdependency

Our tasks were designed to encourage collaboration by prompting children to discuss key questions about the tasks. This was particularly emphasised in the worked examples approach, and not explicitly directed in the step by step resources. They were not designed to deliberately create interdependency between the children to foster collaboration, just to encourage them to discuss the challenge.

The problem-solving tasks themselves also need to be structured conceptually so that they contain opportunities for interdependent work. Problems with multiple facets that are meaningful enough in themselves to be divided would be needed to facilitate interdependence. There are questions around whether children would need to have developed a certain level of skills in programming before they are able to access problems that are sophisticated enough to allow this approach.

Targeting collaboration skills development

Given the time available and the focus of Code Clubs, our resources did not contain discrete or explicit activities to address the development of collaboration skills. The focus of skills development was in the problem-solving space rather than the social space. Although environmental, technology and support factors appear to be more fundamental issues with collaborative problem-solving, it also appeared from our observations and interviews with adults that some explicit skills development would be needed to ensure that children this age have the skills to meaningfully engage in collaborative problem-solving. Task design alone may not be enough to facilitate this complex process.

Which approach is better?

Nesta's collaborative problem-solving framework provides a rich tool for examining collaborative problem-solving in the context of programming and Code Clubs. We have identified the most significant factors in this project being the environmental factors related to technology, classroom logistics, support of adults and a problem designed to rely on collaboration. In our case these are areas to focus on developing first; but there are other aspects of the framework that could be improved and mainly considered through further research.

Although this project did not show a strong difference between the two approaches in terms of collaborative problem-solving, the findings do make clear reasons why this is the case. They demonstrate the importance of creating an environment which facilitates collaboration around computers rather than individual work, as well as the expectations that adults have for the activity that takes place.

Learning new programming concepts

We also explored how the worked example approach affected children's engagement with and learning of new programming concepts. The topic was building new blocks, the Scratch equivalent of defining procedures. We observed the children undertaking the final challenge project where they were asked to draw combinations of shapes, and we interviewed adults about students' engagement with the previous five weeks of tasks.

In general we did not observe a difference between how the two different groups engaged with the final challenge. However, feedback from adults suggested that they had engaged with the programming concepts and progressed.

Observing understanding

Observation of the final task revealed that there is often a difference between children engaging with the programming concepts and completing the set tasks in the informal environments of Code Clubs. For example, some children were encouraged to work creatively by the adult and make their own shape. Having drawn an initial shape, many children decided to explore what they could create with shapes rather than stick to those that were defined for them as the challenge. These children adapted their code for different shapes, demonstrating an understanding of the concepts but not achieving the outcomes set out in the task.

Identifying the changes that needed to be made demonstrates students' understanding of an algorithm. However, observation showed that some of the children were changing values randomly until they got the result they wanted or liked. They sometimes could not identify the part of their tinkering that secured the result and did not remember it for future use. For example, some children struggled to change the blocks of code used to draw a square into blocks that would draw a triangle. They did not know which block would change the length of sides or the angle. One student drew a triangle with sides consisting of curved instead of straight lines. Even though the shape looked right on a first sight, it failed to demonstrate that the amendments to the code were intentional.

Below, we outline the main aspects that seemed to support students' understanding of their code and directed the learning of programing concepts in the desirable way. In so doing, we try to compare whether this was better achieved through the worked example or step-by-step task design.

Ways to improve understanding

Tinkering and experimenting

Tinkering and changing smaller parts of a working program allows students to confirm what those parts are responsible for, and explore how their code works. It allows them to practically try out the ideas they may have and teaches them to explore the possible answers to their own 'what happens if' questions. It can also serve as a useful practical demonstration of otherwise complex and abstract processes that they may find hard to follow and imagine. The worked examples approach supports this by providing a working program and then directing attention to changing key values through questions.

This was observed in several of the clubs. In the final task children were given some blocks they could use to draw a square. Many of them implemented these blocks on screen before tinkering with the numbers to explore how they actually worked. In previous sessions, this could have been facilitated by either the step by step or worked example approach, but worked examples explicitly encouraged it by providing the children with the programs already made, allowing them to focus on adjusting values. Adults described that children built understanding by adjusting aspects of the programs and seeing the results.

It is also important to make sure that the activity relates to programming skills and not just presentation details such as backgrounds and colours. In this example children were exploring what they could achieve with programming constructs by manipulating values. Some adults interviewed told us that this was a particular strength of the worked example approach, as it encouraged children to focus their efforts on the programing constructs rather than on presentation aspects.

"Compared to [step by step projects], worked example works great. It is useful to have the resource ready because even: 'Change the background to…' takes time. And then everyone goes 'Wooo' and then change a colour, and change sounds and… I say: You always have to limit the potential for entropy. You just gotta keep it - still be creative-, but: 'This is the objective, this is how we get there'. So limit the almost secretarial work."

Supporting adults to see progress

A challenge of the worked examples approach that was mentioned by adults in different Clubs was that of the visibility of progress in what the children had created. When children start in a blank programming environment it is quickly obvious what they have achieved in a session to an adult looking at their screen.

With the worked examples projects it is not, as the working program is on screen from the start. This necessitates a more detailed discussion with a child to ascertain what they have learned in a session, although one adult did note that even children following step by step instructions may not have fully understood what they are doing. This is not a major drawback, but it is a perception of adults involved in Code Clubs that should be more explicitly addressed as part of a further implementation of worked examples.

Which approach is better?

We did not observe strong differences in understanding between the different groups on our visits. However, we did gather positive feedback from adults that suggests there may be some in the way the children worked in the sessions before our visit. In summary, both worked example and step-by-step approaches have their advantages and can support different type of learning.

Adults felt that worked examples had been beneficial to the children in allowing them to build understanding better than the step by step approach. The concept of 'reverse engineering' to learn how something works has much precedent, and adults were enthusiastic about its potential for building understanding in Code Clubs.

"I think it was quite good for pupils to have projects that they were mending rather than starting from scratch. When you do that, sometimes you don't get beyond choosing the background and putting the sounds in. So it's good start with having it all there, doing a bit of coding and then I'd say, ok, you can change the background. Otherwise they just get caught up in kind of drawing. "

Adults did raise that projects need to be at the right level of complexity that the children can successfully reverse engineer them.

"The maze one they absolutely loved and they did that one really well. The talking one they loved, as well. They only struggled with the dancing one, because it was a really heavy-coded. They had too many things they were amending."

Our findings suggest that the worked example approach can have a positive impact on the learning that happens in Code Clubs. On a practical level it removes the 'secretarial' work that children often opt to complete before engaging with more difficult programming concepts. It also eliminates some frustration stemming from lower IT skills and instead throws pupils straight to programing. On another level, it allows them to access programming constructs they have not seen before and learn how they work through 'reverse engineering' them and adjusting them creatively to achieve different aims.

The step-by-step approach, on the other hand, gives children the experience of building their own program from scratch; experience that they do not get when the code is partially completed for them. Even though secretarial work cuts from time that can be spent on meaningful engagement with programming concepts, it is necessary part of building a new project; the outcome that clubs often promote. Code Clubs aim to motivate and empower children to get creative with technology, engage with digital making, develop new ideas and create their own projects. Especially in a setting like this, building a project from scratch is thus a crucial skill to have.

One adult expressed that a range of approaches to projects keeps things fresh and interesting, and that this approach would be most welcomed by club leaders and children.

"[A] mix of worked examples and then doing your own problems might work better. The children get to see the possibilities this way, but not sure they internalise the learning in the same way without building something themselves. I would like more of an alternating structure."

Conclusion and recommendations

Summary

At the end of the pilot, teacher confidence and perceived value of all teaching approaches (problem-solving, collaborative and collaborative problem-solving) had gone up (see Figure 4). The biggest increase was seen in the perceived value of collaborative problem-solving approaches and teachers reported a better understanding of what it was and how it could be taught.

Collaborative problem-solving

We compared the newly designed set of learning resources based on a worked example approach to our usual step-by-step approach to find out which one more successfully fosters collaborative problem-solving. Our resources were successful in terms of the problem space as they explicitly targeted the use and development of problem-solving skills. However, despite prompting discussion and cooperation, they did not foster collaborative problem-solving in a strict sense.

The informal ethos and structure of Code Clubs make achieving structured collaborative problem-solving challenging. Fostering collaboration in a strict sense in our groups would require a much more structured approach. This was true for both trial groups regardless of the task design. There are some simple ways to improve collaboration without damaging the informal atmosphere such as requiring children to share computers and encouraging them to work together based on similar skill levels.

Learning new concepts

Alongside collaborative problem-solving, we also examined how worked examples can facilitate students to develop understanding of code and learn new programming concepts, in this case defining a function in Scratch.

We found that worked examples could encourage children to take creative approaches and tinker with code more than step by step instructions. The types of worked examples we used included questions to focus the children's attention on tweaking particular aspects of the code and developing their understanding of how these sections are working. This approach means the limited time children have in Code Club can be spent understanding concepts rather than having to build a program that contains these concepts from the start. More time spent on a concept is likely to result in more secure understanding of it.

Adults running Code Clubs were positive about the worked example approach, seeing its potential for developing understanding of complex concepts. Those using step by step instructions said that there was a need for projects that built understanding in the way the worked examples projects did. However, adults using worked examples did say it was harder for them to monitor and understand how children had progressed with a project quickly when asked to help them. This was because they could not judge this based on the completeness of the program they saw on screen, which would have been entirely created by a child and therefore easy to judge using the step by step approach.

Tinkering and experimenting with existing code takes away part of the complexity and allows children to focus on smaller parts of a problem. Thinking about how to change code to reach different outcomes can make children reflect on the effect of different parts of the code. Nevertheless, it works best if a task allows children to see the effect of their amendments immediately.

Worked examples were welcomed by educators and appear to be a valuable addition to Code Club learning resources. They could work especially well in combination with a standard step-by-step approach. Step-by-step instructions lead children through the whole process of creating a project, from start to finish; the experience that is essential if motivating children to create their own projects is the aim. Worked examples, on the other hand, could work well to break down problems, unpack more complex concepts and direct learners' attention on the key aspects of a task. We plan to developed more learning resources based on worked examples for use in Code Clubs interspersed with step by step projects, and focused on key concepts that might be more challenging or at points when they are particularly new to children.

Key findings

- Fostering collaborative problem-solving takes a structured approach, and needs to be closely facilitated, particularly in informal learning environments. This might involve setting up the environment to strongly encourage collaboration, or explicitly giving children roles to take.
- Worked examples based resources can encourage and allow space for children to take an exploratory and creative approach to programming.
- Worked examples can focus children's attention on key aspects of the learning objectives, compared to the step-by-step instructions where their focus can end up on other aspects of building the project such as presentation.
- It can sometimes be difficult for adults to see the progress with worked examples since the children hadn't built them from scratch.

Next steps for the Raspberry Pi Foundation

- Put in place strategies for volunteers to facilitate collaboration such as children sharing computers and guidance for volunteers.
- Develop more learning resources with a worked example approach, particularly for more complex concepts to help children focus on mastering them.
- Adapt the worked examples approach where appropriate so children make amendments to the projects to demonstrate progress.

Bibliography

Kagan, S. and Kahan, M. (2015) 'Kagan Cooperative Learning.'

Luckin, R., Baines, E., Cukurova, M. and Holmes, W. (2017) 'Solved! Making the case for collaborative problem-solving.' [online] Nesta. Available at: https://www.nesta.org.uk/sites/default/files/solved-making-case-collaborative-problem-solving.pdf [Accessed 13 Apr. 2018].

Sweller, J. and A. Cooper, G. A. (1985) The Use of Worked Examples as a Substitute for Problem Solving in Learning Algebra. 'Cognition and Instruction.' Vol. 2, No. 1 , pp. 59-89. Available online at: http://www.jstor.org/stable/3233555 [Accessed 13 Apr. 2018].

Learning resources

Learning resources used in this project and referred to in the report are available online at rpf.io/cpsresources

58 Victoria Embankment
London EC4Y 0DS

+44 (0)20 7438 2500
information@nesta.org.uk
@nesta_uk
www.facebook.com/nesta.uk
www.nesta.org.uk