Assessing Computing

By Phil Bagge

Computing Inspector/Advisor Hampshire

CAS Master Teacher

Teacher Calmore Junior, Ringwood Junior

And Otterbourne Primary Schools

Author "*How to teach primary programming using Scratch*"

code-it.co.uk resources

Talk for Computing

My Journey so far on my continuing mission to understand learning in computing

My *"Research"* Base

- An Advanced Skills Teacher of ICT since 2007
- Teaching *organised* Computing Science at six schools since 2011
- Over 1800 hours of Computing Science taught at 6 schools

Discoveries so far

- Primacy of Computational thinking in programming
- Unplugged (never leading to programming) great as introduction to concepts in primary but don't get stuck there
- Balance between instruction and exploration in programming is important
- Computing Science fantastic for promoting resilience
- Don't start with a fully formed assessment system or you set expectations too low

Resilience in Computing spreads to other curriculum areas

- Video can be found at
- Ringwood Y5 Video

Computing helps pupils solve problems in other areas of the curriculum

- Ringwood Y4 Video Problem Solving

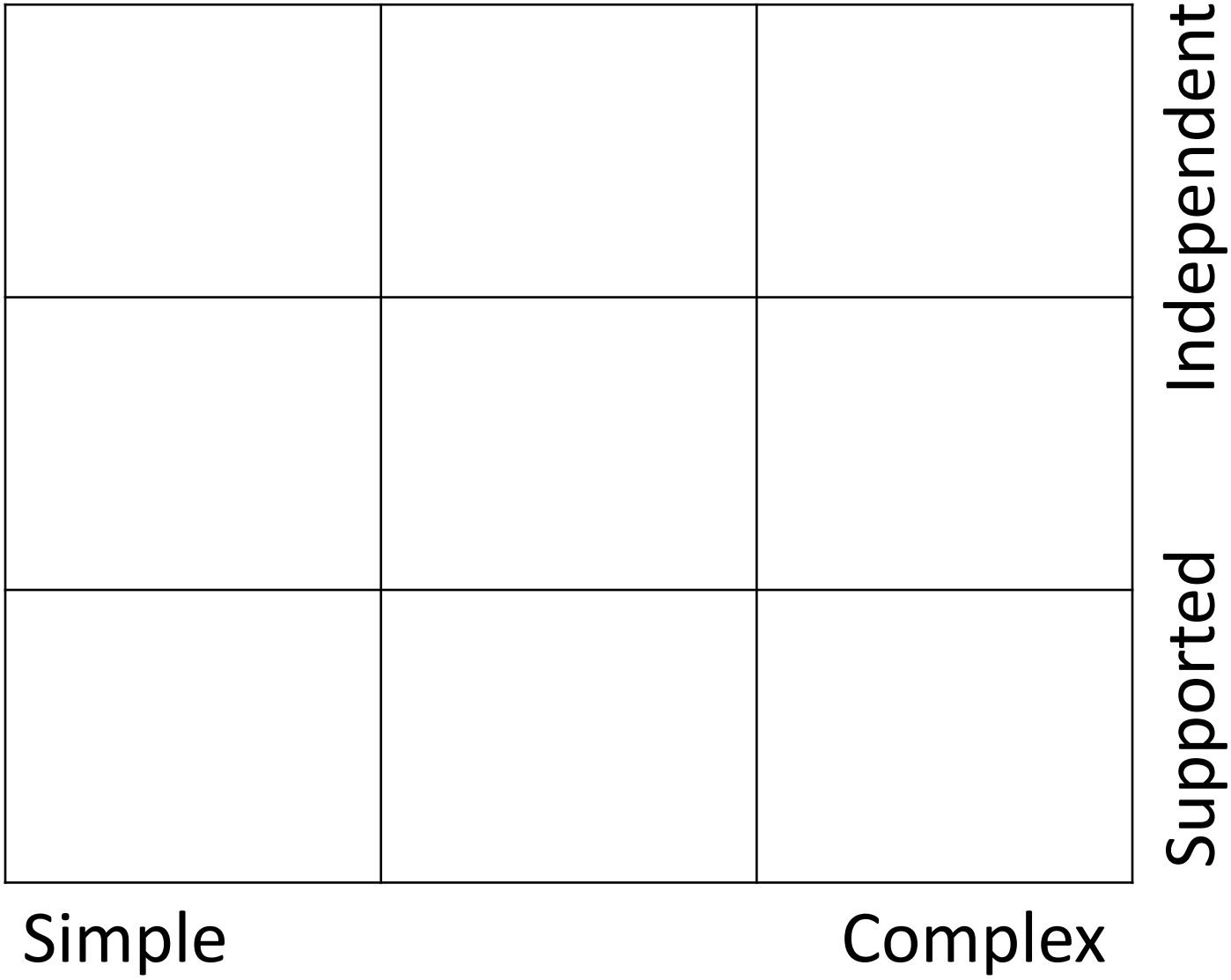
Through emphasis on pupils solving problems themselves I have more time in my lessons

….. More time to capture the learning

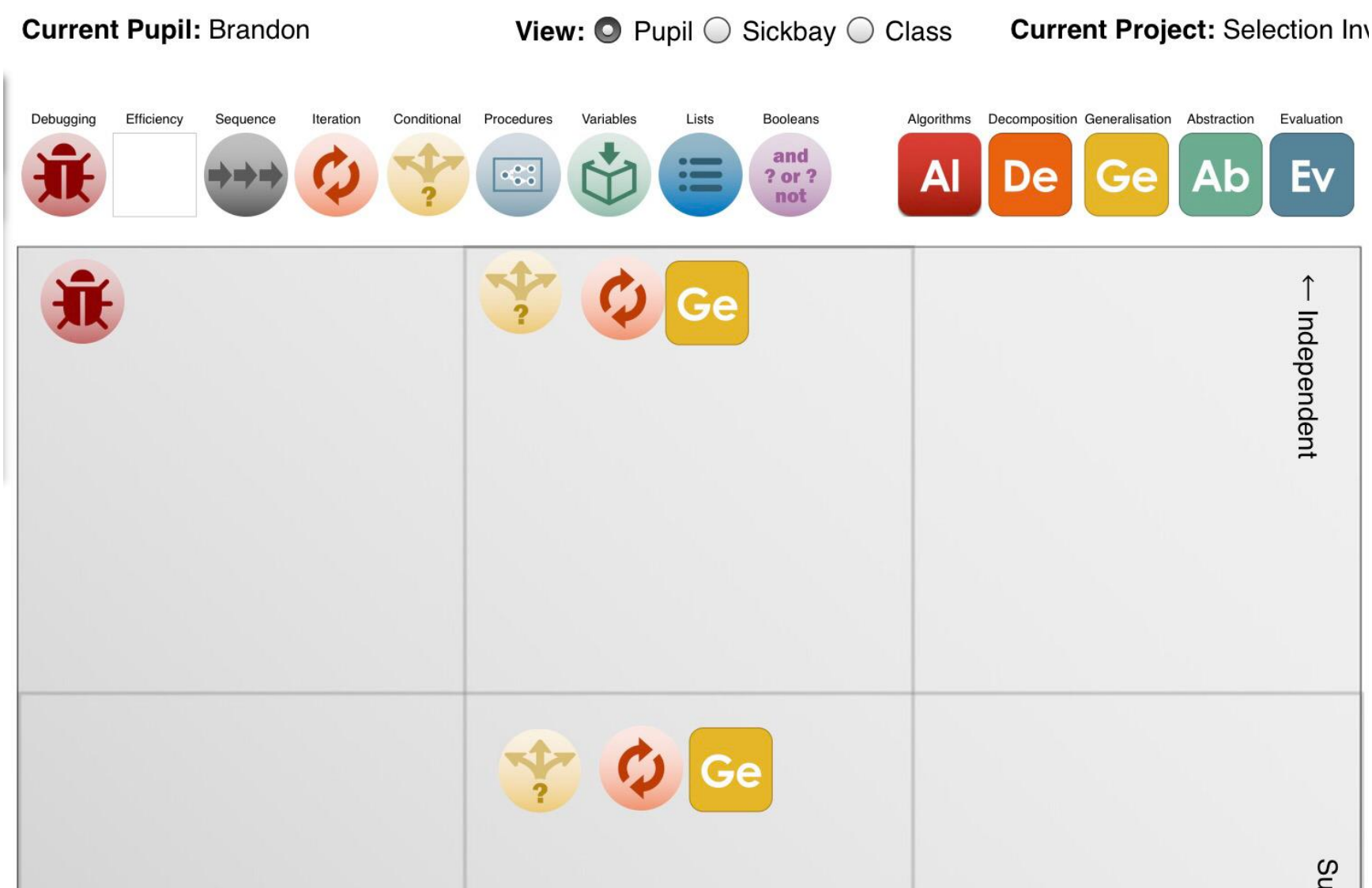
What am I attempting to capture and assess?

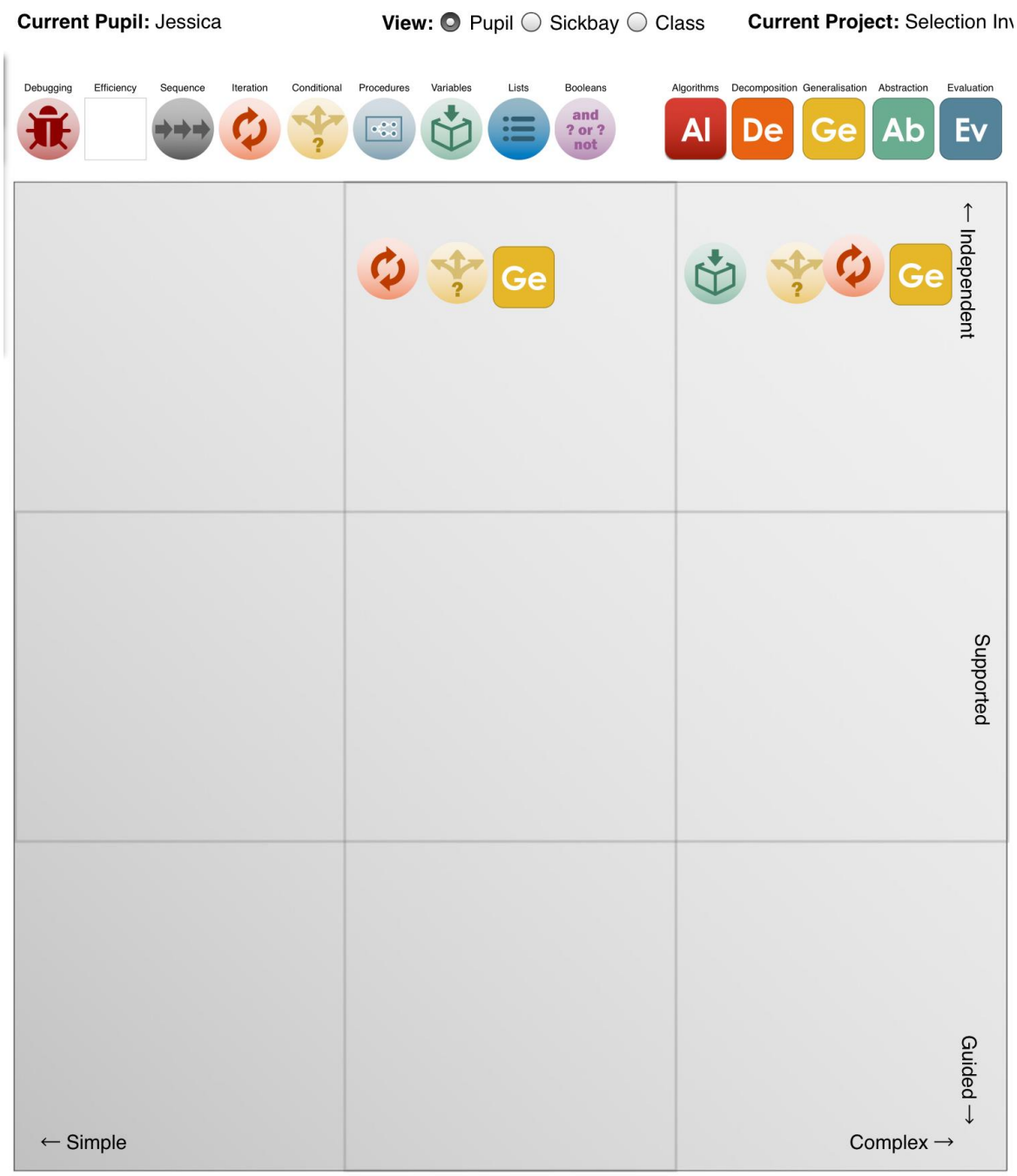
- Computational thinking in action
  - Algorithmic thinking & efficiency
  - Decomposition
  - Generalisation
  - Abstraction
- Programming Understanding
  - Sequence, Repetition, Selection & Variables
- Independent or Supported
- Complex or Simple
- Debugging

The Grid



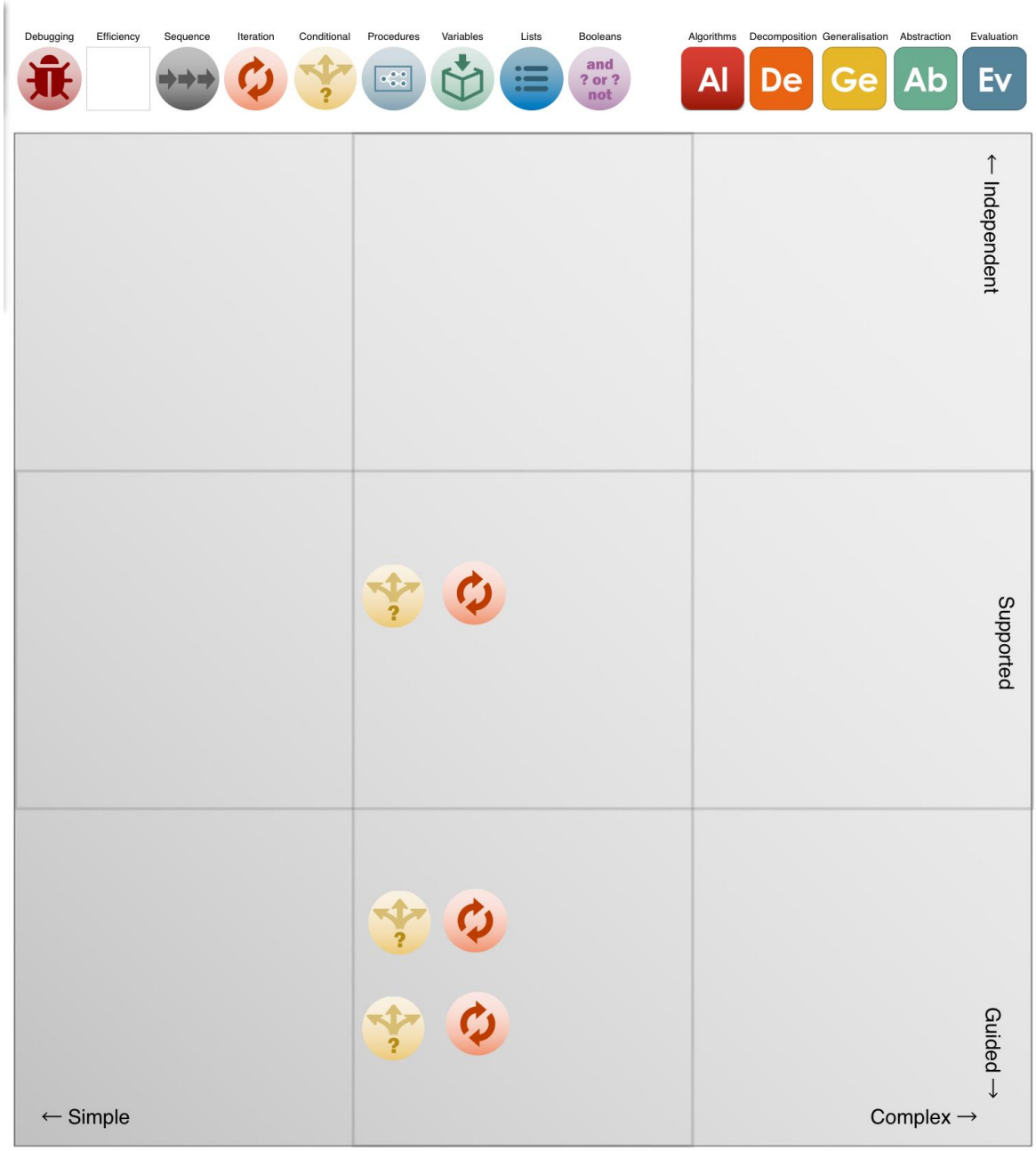
Assessing Computing in Lessons





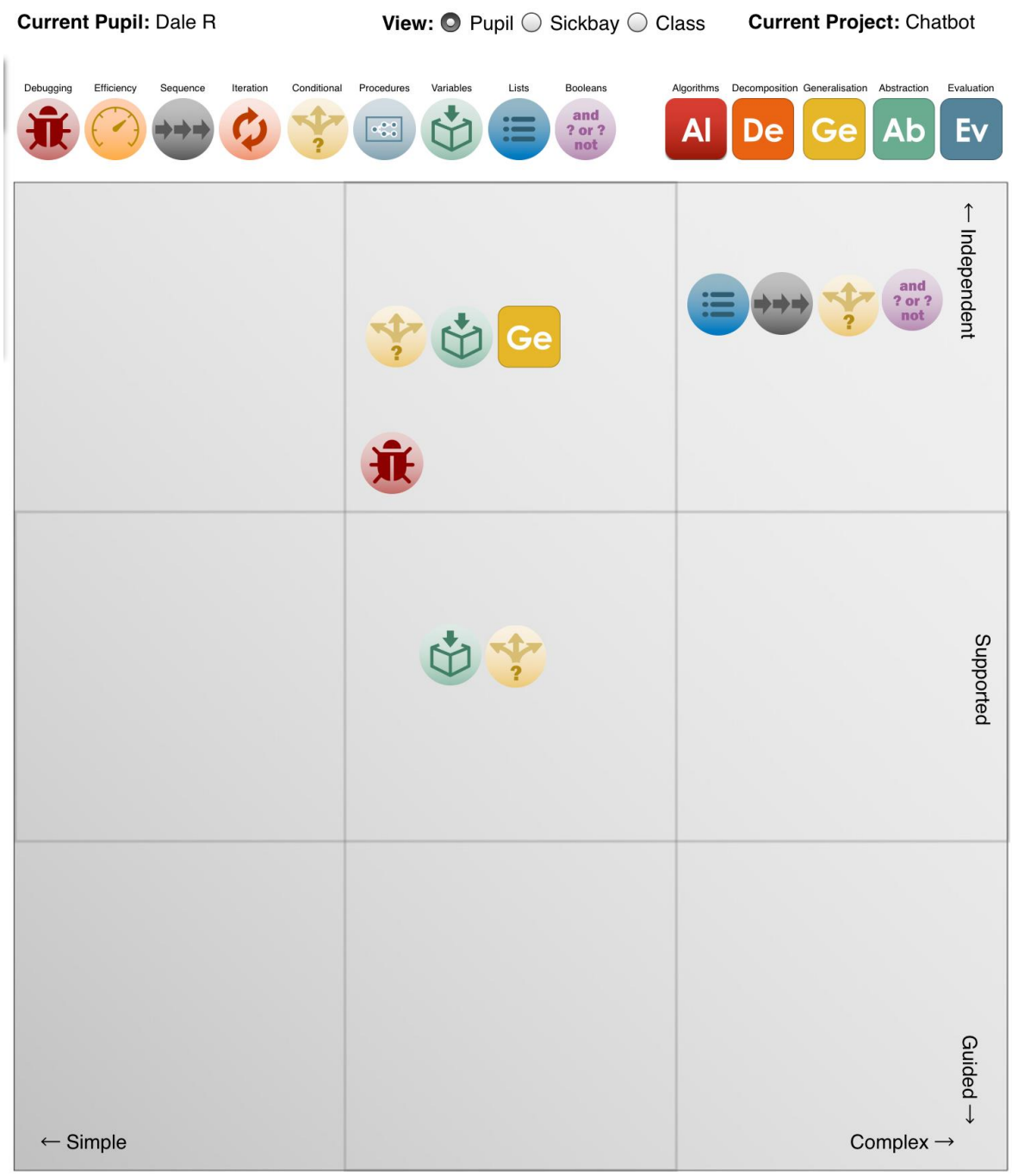
When training other teachers I have got them to discuss pupils programming projects and use this to record pupil progress

It helps to discuss complexity and independence



Talk for computing allows you to ask pupils if they had help and take this into account when assessing their work

Here a pupil has had considerable support from their neighbour discovered through discussion with teacher

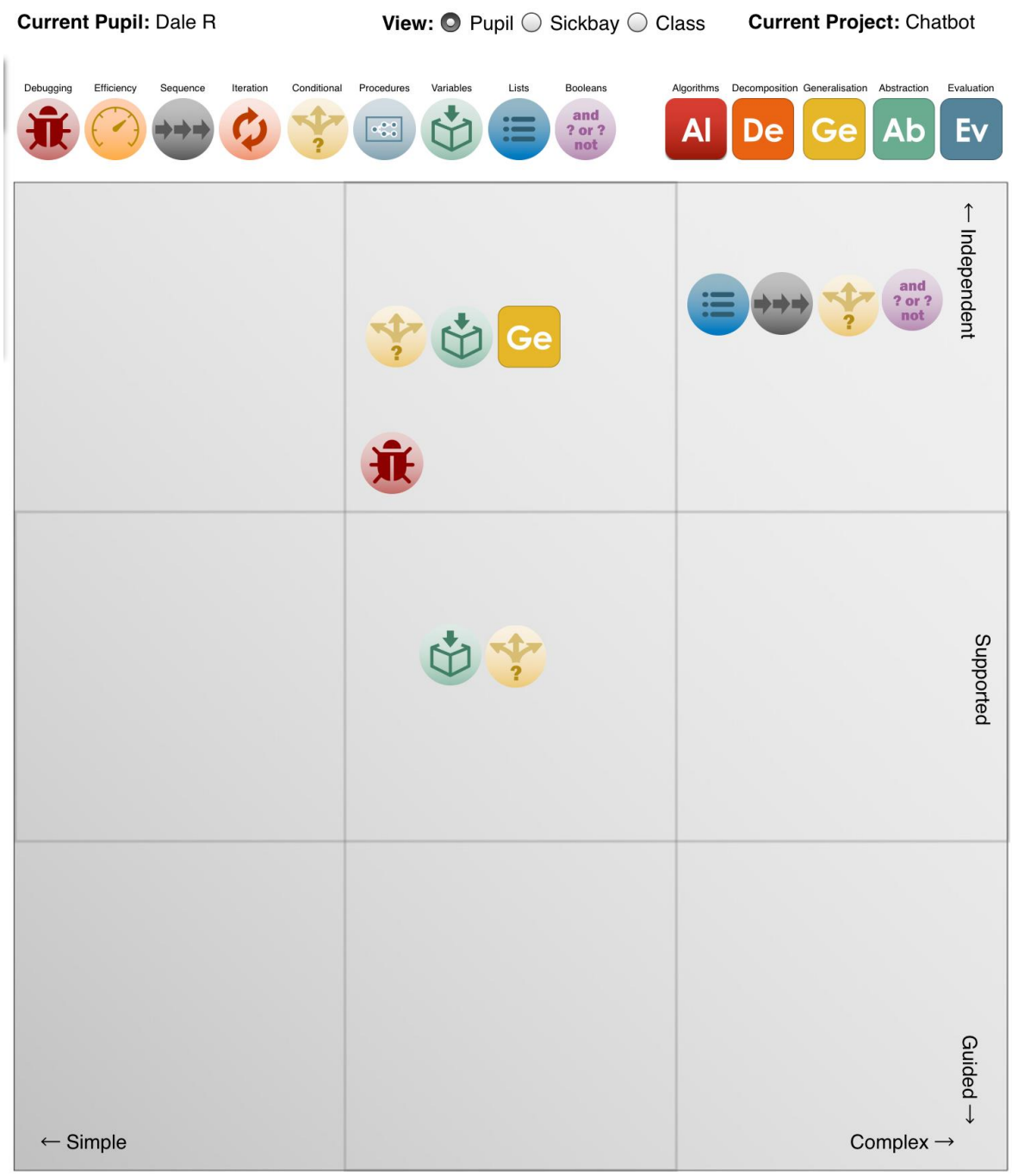


Deciding what is simple and what is complex encourages me to adapt my own progression grid

Algorithm	Algorithm (programming) Evaluation	Decomposition	Generalisation	Pattern Spotting	Abstraction	Design	Sequence	Selection	Repetition	Variable Use	Input Output	Debugging
Think of a simple everyday algorithm (Remove toothpaste top, squeeze toothpaste onto brush)	Identify if an algorithm does what you want it to do (Does Bee-bot reach its target?, Is Scratch conversation easy to read?)	Break a simple everyday algorithm into parts (breakfast, getting changed, walk to school)	Recognise where an idea is adapted and used again as directed by the teacher (Perimeter)	Spot simple patterns in code or algorithm (simple regular 2d logo shapes)	Define all the elements in something and then remove the ones that are not needed (Music as code)	Following instructions to create (Smoking car, Music machine, dressing up game)	Follow simple everyday sequences of instructions	use single simple condition (playground games flowchart decisions)	Explore and use simple repetition in music and dance	variable used to hold number or word and reported (Quiz, Counting machine)	simple inputs (keys, mouse click, switch etc) control on or off (Lego Wedo toilet fan, car park barrier)	Recognise that there is a problem say what problem is
read and follow symbol sequence algorithm (PE Cards, jump, step etc)	Recognise that there are more than one algorithm to do the same thing (Identify two bee-bot routes to go from A-B)	Observe a working program and decompose its elements as a class (Smoking car, slug maze)	Pupil chooses to Adapt ideas that they have used to solve similar problems (Selection investigation, primary games maker, Random word)	Spot patterns in algorithm or code and continue the patterns (Coin machine, program a clock)		Add small non critical adaptations (Smoking car, Music machine, dressing up game)	Create simple everyday sequences of instructions	create selection within a loop (Slug maze, selection investigation)	Create simple repeat x times loop (Scratch Jr, scratch music notes in repeat loop, repeat x in logo)	multiple non connected variables used	Changing state other than on off such as fast slow bright dim etc (Lego Wedo toilet fan, car park barrier)	identify where in the code or algorithm bug/ problem occurs
Create simple sequence algorithms using symbols (Bee-bot cards, human crane cards)	Recognise that one algorithm may be better (Bee-bot shorter distance, less time. Less algorithm symbols)	Observe a working program and decompose its elements as an individual (Tables Game)				Adapt a given design for a new teacher given purpose (Times tables game)	create sequence of symbols (Bee-bot cards, human crane cards)	Use single maths operator condition (Maths quiz)	Create non terminating continuous (forever) loop (making sprite move slug maze, crab maze)	variables that change inside a loop (Counting machine)	Use sensors to control or report (Lego Wedo tilt switch)	Debug simple sequence errors independently
Read and follow written sequence algorithms (Forward 3, right 90)	Design an algorithm/code for a specific person or group of people (design a Scratch program for younger pupils)	Create a program by decomposing it into parts and solving parts separately (Primary games maker)				Repurposing ideas for a pupil chosen purpose (Selection investigation, Primary games maker)	create sequence of simple code that can be easily read (Scratch Junior blocks, say or think blocks in Scratch)	Multiple selection beyond if and else (Scratch use of multiple ifs perimeter planning)	loops within loops for a reason	variables interacting with other variables (train computer to do maths)		Debug simple repetition, selection & variable errors independently
write simple sequence algorithms using words (Jam sandwich algorithm) read and decode musical algorithms (Music Score) use rules algorithms	Evaluate more complex code that does the same thing (Logo design with procedures and without, Scratch code with loops and without)						create multiple sequences running concurrently (Scratch music machine notes and beat in separate blocks)	Multiple conditions using AND OR NOT (odd and even)	Create and use loops that terminates when condition met (repeat until loop in coin program)			Debug repetition, selection & variable errors independently
Read and follow algorithms with selection and repetition (Playground algorithms, number machine)							create sequences or multiple sequences where timing is critical (control sprite on times tables game, conversation sequence)					Dividing up code to find where the error is or running Scratch sets of blocks separately
Complete unfinished algorithms with selection and or repetition (Coins sorter, Program a clock)		**COLOUR KEY**	MAINLY KS1 ONLY	KS1 & KS2	MAINLY KS2 ONLY	KS2 & KS3						
Create an algorithm with selection or repetition									code-it.co.uk			

Primary Computing Science Progression

http://code-it.co.uk/wp-content/uploads/2015/08/progression_gridv2_codeit.pdf



If you wish to try the web app you can

http://users.ecs.soton.ac.uk/lac/csasses/v6/hias.html

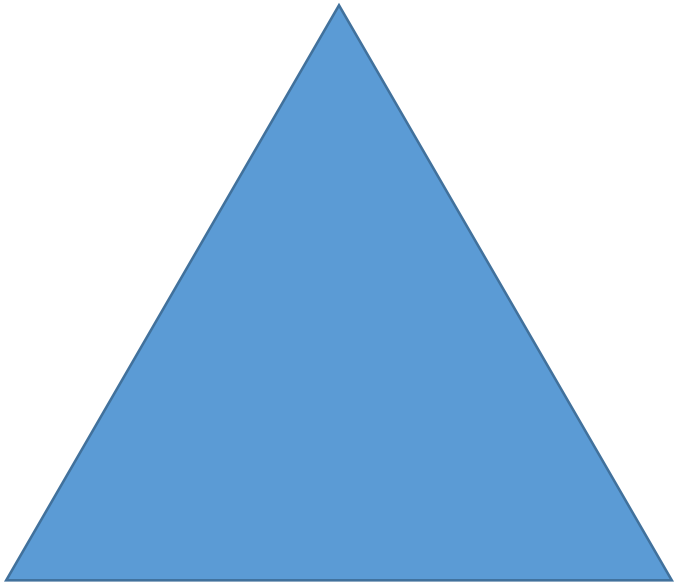
All data is held in browser storage

Thanks to @lescarr
For all his hard work coding the app

I discovered more through dialogue with pupils than I did just assessing their work outcome

Talk for Computing

Talk for assessment with teacher app



Talk booth

Talk in class

Talk for Computing Video Booth

Good Questions

- Most of time questions need to be directly related to what they are working on
- Framed to draw out computational thinking not always mention it directly (may be different with older pupils)
  - Unless a word is in general school use is part of the schools vocabulary not always useful to use it
  - Can you describe where you have used computational thinking skill of generalisation in your programming project? OR
  - Can you explain where you adapted an idea that you used somewhere else?

Year 3 Reflecting on Decomposition

- Decomposition Video Not Available Online

Year 3 Describing their bugs

- Bug Video Not Available Online

Year 4 Reflecting on Generalisation

- Generalisation Video Not Available Online

Talk for Programming

Class reflecting in computing suite

- Coding Video Not Available Online

Talk for computing logistics

Booth

- Green screens great as background
- One take only especially with older pupils
- Compulsory not an option

In Class

- Small stand is important for sound and hands free
- Some lessons it is about choice and others it is compulsory

Digital Literacy skills are important

- Framing face or screen
- Knowing where microphone is
- Everyday use not 'super special'
- Saving videos

Practicalities Learnt

- Pairs in the booth don’t work unless they have predefined interviewer interviewee roles
- I’m not the best person to carry out the interviews for assessment
- Can’t just rely on volunteers everyone needs to be forced to have a go
- Teach some digital literacy videoing skills first

Where next?

- Talk for maths, literacy, history etc
- The grid in other subjects or topics
- Reflecting on teaching and learning
- Developing a great questions bank
- Summary of response

Come and chat
Phil Bagge
@baggiepr
code-it.co.uk

'How to teach primary programming using Scratch'
http://goo.gl/W4bQ1a

★★★★★ **What a great range of coding task books**

By Matt Warne on 10 Jun. 2016

Verified Purchase

Phil Bagge does it again - an out the packet book that structures coding concepts for primary pupils. Great success criterias for progression on tasks. These books are a one stop shop for primary computing teachers out there. I have written a quality scheme of work from his progressive tasks and my pupils are making astonishing process. I recommend all books in this series!

Thanks again Phil - quality book with all resources to print for pupils