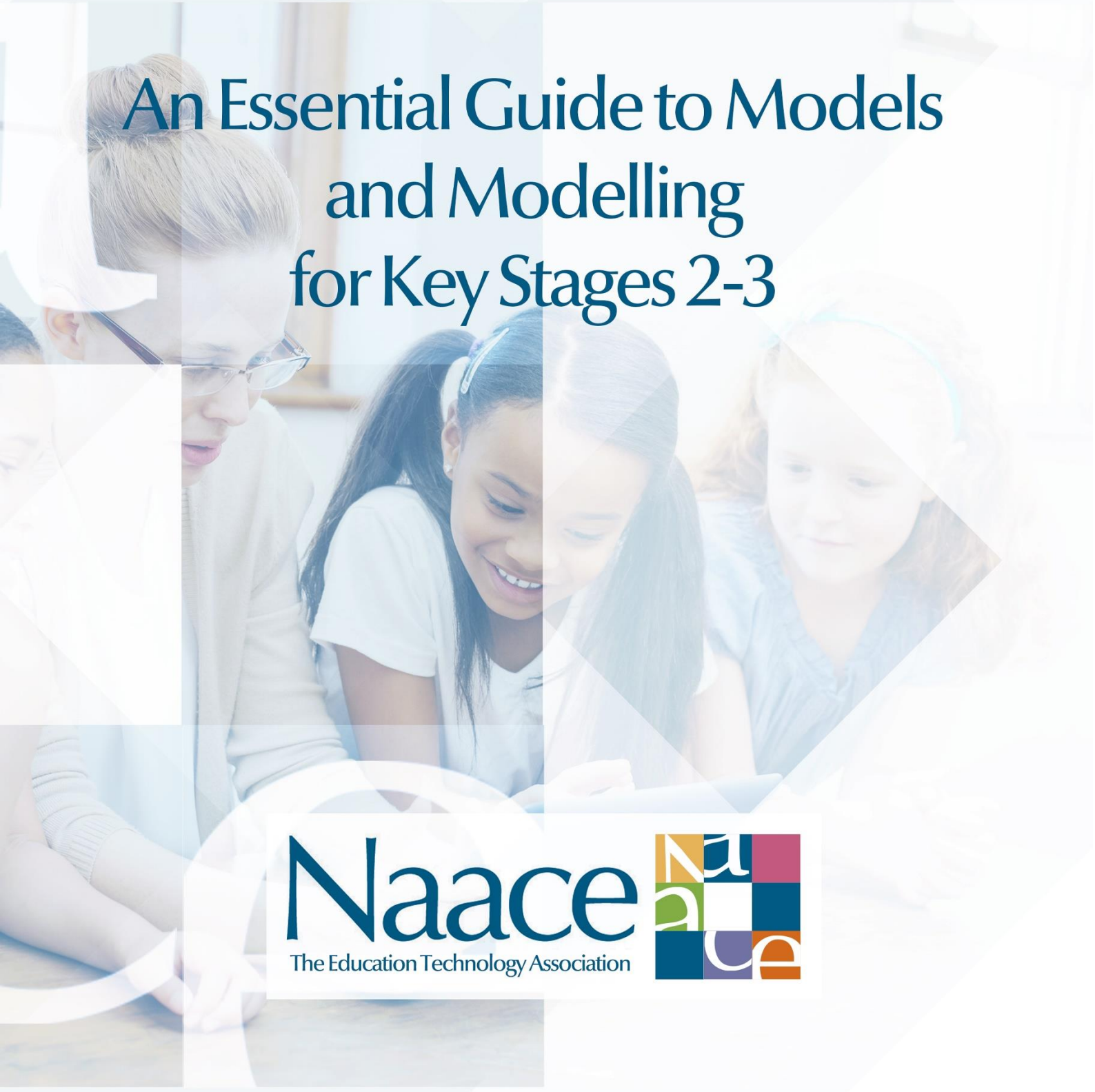


An Essential Guide to Models and Modelling for Key Stages 2-3

Naace
The Education Technology Association

A Naace Publication by Tim Scratcherd
Series Editors: Mark Chambers, Dr Carol Porter, Tim Scratcherd

About the Author

Tim Scratcherd is director of Learning Linked. Contact him via tim@learninglinked.co.uk

Summary

A model is a representation of some part of the real world. There are two sorts of simple model; models of objects, and models of systems. An everyday example of a model which is an object is a portrait. An everyday example of a model of a system is weather forecasting, where changing weather data is gathered, and mathematics used to predict further changes. Models are fundamental to many aspects of life.

Building models involves mapping aspects of the world to a modelling context. For objects which are models, the object should look like the real object. For systems models, the model should have interactions which work like the world.

Here we describe four different types of systems models; models based on randomness, deterministic models, financial models, and qualitative models. Making models by programming is a specific requirement, and we will describe program based approaches to these different types of model separately.

Introduction

Systems models have all kinds of uses. There are practical ones, such as finding your way (Google Maps), knowing when to cut the grass (the weather forecast), and the likelihood of your business being viable. Games which reflect some aspect of the behaviour of the real world are often models for the purposes of entertainment. Models have many scientific and experimental uses, especially when it is very difficult to experiment with the real thing, such as the behaviour of galaxies.

In England, the place of modelling in the Computing Programme of Study is most apparent in Key Stage 3 where it is stated that pupils should be taught to:

- design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems

which is a specific statement of the requirement to build systems models based on programs. Systems models abound in another strand of Computing, Digital Literacy, and the underlying concept of a system is key to a third strand, Information Technology.

Being able to understand, use and build models is a high level capability, and if you have it, you are empowered both as a learner and as an individual.

It is not enough for a student to know examples of models, and to know roughly how some of them work. It is not enough for students to be able to use models. Students should also be able to build their own models, choosing the right type and approach, and ensuring that the models they make are fit for purpose because they work like the world.

Key Concepts

What is a **system**? In the modelling sense it is something which has a **purpose**, or **function**. To achieve this function, it will have **outputs**, which will require the **processing** of **inputs**.

An everyday example is that of a washing machine. Its purpose is to clean clothes. You might like to try and work out what its inputs, processes and outputs are before reading on.

- Its inputs are (at least) dirty clothes, detergent, clean water, and electricity
- Its outputs are (at least) clean, wet clothes, dirty water, noise and sometimes light
- Its processes are whatever program you enter.

Note that you also need to set it going. Another example is that of a green plant. In daylight, if it is warm,

- Its inputs are light, water and carbon dioxide
- Its process is called photosynthesis
- Its output is glucose.

So in daylight its function is to harvest energy from the Sun and store it. The fact that it can do this is the basis of most life on Earth, through food chains. At night time, it is like any other living thing; we leave you to work out its inputs, processes and outputs then.

The approach, of considering inputs, processes and outputs for existing systems and systems you might want to make, is called **systems thinking**, or even **systems analysis**. You might like to experiment with applying systems thinking to other objects and situations, such as a home, or a town, or a laptop.

Note that systems do not always have to have inputs, apart from the energy to drive them. Consider the Solar System, and the Water Cycle. These systems are often called **closed**. Once started, they just continue to process (we hope), although they must have had **initial conditions** to get them started.

A system does not have to be a model; indeed both the examples of closed systems are not models at all. They are the real thing. A system becomes a model when there is a **mapping** from **components** of the real thing to components in the system and **behaviours** (or **rules**) of the real thing to behaviours in the system. The term **computational abstraction**, taking from the Programme of Study, simply means the component in the system which matches the component in the real world. This will become clear when you read the examples given in the sections on each model type.

Many models are **opaque**, in that you can't see how they work. You can't check out or change the mappings. All you can do is change the inputs and then observe what happens with the outputs. This by itself is a very powerful thing to do, if the model is accurate; it

helps understand how the real thing works. Models like this are called **simulations**. Have a look at Google Earth.

Transparent models allow you to see and modify their workings. This provides the opportunity to change the rules inside models.

There is a fundamental difference between building a model just for you to use, and building a model **for other people to use**. If other people are going to use a model, then there will also be a need for instructions, and inputs, which allow the user to interact.

Interactive models allow a user to make changes in the inputs, and observe the effect in the outputs.

To modify existing models, or to build models from scratch, an understanding of the **environment** in which the model will be built is required. This environment must provide mappings for both the components and the behaviours. Several environments will be used in the examples following, which are sufficient to illustrate the key concepts.

Further examples of models of all four types and from several environments, are available on application.

Models based on Randomness

Why do we toss a coin at the start of a game of football or cricket, allowing one side to choose the initial conditions? This is because it is assumed that we cannot predict a head or a tail, but both are equally likely. Is this true? One way to test it is to build a model, which allows us to spin many coins, and look at what happens. Excel 2007 onwards gives us a great environment to build a model. The function

=RANDBETWEEN(0,1)

When inserted into a cell, generates either a 1 or a 0. Both are equally likely, but the next value is not predictable. This allows a very simple mapping of component, a coin to a cell, which contains the formula, and we can assign a 0 to heads and a 1 to tails. To set up the behaviour of tossing the coin, the sheet needs to recalculate. This can be done very simply by pressing function key 9 (F9). This completes the mapping and the model is built.

- Coin <-> cell containing formula (this is the computational abstraction)
- Coin toss <-> recalculate

Try this out by copying the formula into a cell in Excel, and holding down F9. Watch how the value in the cell varies unpredictably between 0 and 1. It can be extended in many simple ways, which also have the advantage of introducing transferable skills with Excel.

Begin by modifying the output to improve its mapping, by showing heads and tails instead of 0 and 1. Use the formula

=IF(RANDBETWEEN(0,1)=0,"heads","tails")

Simply copy and paste the formula into a cell in Excel, press F9, and watch what happens. This also shows how the IF function works.

To get more coins, simply keep copying into new cells, or replicate the cell in the usual way. Start with ten coins. Press F9 a few times, but watch what happens. Notice that getting exactly five heads and five tails is uncommon. So what does it mean to say that tossing a coin is fair?

To continue this investigation, you need to make a lot of coins, and find a way of adding up the heads and tails. Randomness is a fundamental way that the world works, and a mastery of modelling randomness will help model many real world situations, from very simple ones such as dice, rain falling and water freezing, to radioactive half-life.

Deterministic Models

There are three sorts of deterministic models. The first is where the behaviour of the model is determined by one or more equations. Here is a simple example.

A frog is 220 cm from its pond, when suddenly it is afflicted by the well-known hopping sickness, which means that it can only hop half as far next time as it did the last time. Its last hop, to where it is now, was of length 200 cm. If it heads straight for its pond, does it get there?

This problem needs only a calculator to investigate it. Use the calculator to continue the table below.

Next hop	Total distance gone after next hop
100	100
50	150
25	175
12.5	187.5

To make a model, use a spreadsheet such as Excel.

In A1, put 100. In B1, put 100. In A2, put =A1/2. In B2, put =A2+B1. Then replicate down from A2 and B2. It should not take long to notice that the total distance hopped approaches but never exceeds 200cm. So the frog can never get home. See this at https://youtu.be/IyuqaMHrlGg

Charting just column B (mark the column, insert>chart>scatter) gives a visual representation of this.

To turn this into a model to investigate what the hop size must be for the frog to get home, create a store for the hop size in, say, C1. Put 0.5 in C1 in the first place. Change A2 to =A1*C1 and replicate down. This allows someone to experiment with hop sizes and is a very simple example of making a model for someone else to use. You tell them to change the value in C1 and observe what happens. See this at https://youtu.be/yA2p4Nh5MM4

Examples of deterministic models driven by formulas abound in applied mathematics and physics. Because of the mathematics involved, many students will find making such models difficult. They can still be shown models. Models available on application are throwing a ball (projectiles) and simple harmonic motion.

The second sort of deterministic model is one where a sequence of values is obtained, and this sequence of values is chosen to represent some key aspect of a feature, and the values model those key aspects, and yet there is no mathematical formula. These are empirical deterministic models. Very common examples relate to mapping; contour lines for ground maps, and isobars for weather maps. There is of course a fundamental difference between these two examples; one refers to an aspect of an object (the ground height) and the other to a system. Students will also encounter modelling when data logging in science.

The last sort of deterministic models are simulations. http://solarsystemscope.com/ is an impressive perspective view of the planets of the Solar System in orbit round the Sun, which allows the running conditions to be changed but not the workings of the model itself.

Financial Models

Three sorts of financial models will be described, all of which are most conveniently worked in a spreadsheet. The simplest is where the costs of items are **compared**, such as a basket of shopping from different supermarkets. Students might gather the prices for the same collection of items from different sources, sum them, and discover which is cheapest.

The next sort of financial model is one where there is a **budget**. A typical example is that of planning for a party. Students might be given £500 and asked to plan a special occasion for a given number of guests, with food and entertainment.

The third sort of financial model is one where there is **income** and **expenditure**, bringing in the additional concepts of **profit**, **loss** and **break even**. Students could be asked to plan a profit making event for charity, where a band is hired, and there is an entry charge. One of the interesting aspects of this sort of financial modelling is that it provides an opportunity for **goal seek**, where, once the model has been set up, the minimum number of paying customers can be found to ensure break even, i.e. neither a profit nor a loss.

In practice, it is not enough, when running a business, to ensure that there is ultimately a profit. Real businesses always have a lag between income and expenditure, in that you have to spend money to buy goods or ingredients before they can be sold. To cover this, businesses need **startup** finances, and must predict their **cash flow**, which is the tracking of income and expenditure over time.

In all of these models, there are opportunities for students to make models either for themselves, or for others to use, where users simply vary the inputs by changing values in key cells. For these a guide or manual will be essential.

Qualitative Models

These are models where arithmetic is not required! There are two sorts of qualitative model; topological, and sequential.

A **topological** model is one where the pathways through the model accurately map to the real world. A common example is a virtual visit or tour, often made by secondary school students as an introduction for potential new students, to defuse the feeling of getting lost. Building such a model entails identifying locations such as classrooms, and the routes which connect them. In the model locations can be shown visually by photographing them, and the routes by hyperlinks between elements of the images. The best environments for building models like this are multimedia software environments, but presentation packages which support hyperlinks will do, such as PowerPoint.

A **sequential** model is one where a sequence of events is visualised, generally by an animation. Here the feature which turns the animation into a model is the accuracy of the mapping to elements and behaviours, and if there is no user interaction, then the model is a simulation. A simple simulation of this kind is of the passage of a boat through a canal lock. See https://youtu.be/rsNzDJw_p7I. Note that the mappings are accurate.

Another interesting form of sequential simulation is a **chronology**. Here, a sequence of events is displayed at a rate proportional to the actual events. See https://youtu.be/NaTUyk6ndfA and note the comparatively long time before the king's second marriage.

Models and Programming

It is possible to find programming environments for modelling in all four of the model types. However, programming approaches are particularly suitable for modelling using randomness, and for deterministic models. Within that, some languages are more suitable than others dependent upon the model. We recommend that students encounter at least four environments; Logo type, blocks type (such as Scratch), html, and a basic type such as Python. For more detail on this, see the Naace publications on programming. Python and Basics are particularly good for randomness; block languages like Scratch particularly good for any visual models. Html can be used for multimedia modelling, because links between pages can be made; this is not straightforward manually, but by using a software package which uses html as source, such as Dreamweaver, something of the best of both environments can be obtained.

It is important to note that the same principles apply to model building as before. The process is to make a mapping from the real world to the features of the environment outputs and inputs identified, and then only at the end doing the **coding**, which is the sequence of instructions specific to the environment. A very simple example is that of building a model of an ornamental fish tank with fish, in Scratch. The mappings are

Fish<->fish sprite
Tank<->suitable background
Fish swimming <-> sequence of instructions to move the sprite

No inputs are needed; the output will be the fish sprite moving appropriately against the background.

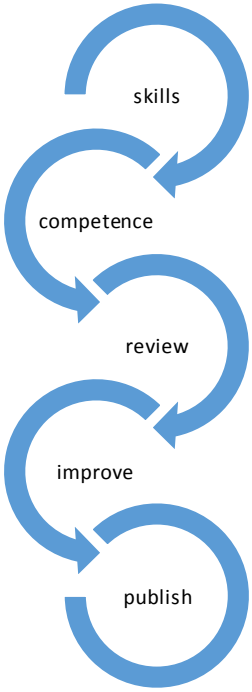
Key Outcomes

There are three modes of working with models, which students should encounter. These are using a model for learning, changing a model to achieve new outcomes, and building a model from scratch.

You should consider developing a map of the kind below, which may be used to plan a student's experience.

Model type	**Random**	**Deterministic**	**Financial**			**Qualitative**	
			Compare	Budget	Income/expenditure	topological	sequential
Interaction mode							
Use							
Change							
Build							

Key Methodologies



Prospective chefs are not expected to learn their craft only by listening to lectures and reading recipe books. Similarly, students need to do more than passively watch demonstrations of how to use packages and programming languages.

So how should they learn?

Clearly, there needs to be some direct skills teaching, and this is best done within the context of setting a problem to be solved. A problem within each of the model types should be identified by the teacher, and a model written. Students should then be encouraged to use the model. At all times be open to learning from the students; they often know far more about package and language functionality that you haven't had time to explore, so embrace their knowledge and expertise.

Once a set of skills has been mastered, students need to develop competence in their deployment. Let them experiment with an existing model. Can they develop or extend it?

Self-review should be an ongoing process. Encourage positive and constructive peer-review by projecting student work-in-progress for discussion. Having listened to the feedback, students are able to make informed decisions to improve their work further.

The essential feature of all review should be the accuracy of the model. Does it behave sufficiently like the real world? This is the acid test. All too often, students miss the point of modelling. We have seen too many sharks in models of fish tanks in Scratch. And scuba divers being chased by sharks. This blurs the distinction between a **game** and a model.

When ready, the finished artefact should be published. Confidence and competence are of course different, but the growth of each is usually linked to the other. Confidence often comes with positive praise from a genuine audience. Celebrate your students' growing competence by posting their work on your class blog, or photograph the work and Tweet about it.

The sequence of skills acquisition, develop competence, review, improve, publish can be followed with any strand of the Computing curriculum, not just modelling, with any new set of skills, and with any age group.

Teaching Resources (bibliography, web links, teaching tips, examples)

Canal lock simulation: https://youtu.be/rsNzDJw_p7I

Download Google Earth from https://www.google.co.uk/intl/en_uk/earth/

Frog hop models: https://youtu.be/IyuqaMHrlGg
Henry VIII and his wives: https://youtu.be/NaTUyk6ndfA

http://solarsystemscope.com/

https://youtu.be/yA2p4Nh5MM4