The BehaviourComposer 2.0: a web-based tool for composing NetLogo code fragments

Ken Kahn, kenneth.kahn@oucs.ox.ac.uk
Oxford University, 13 Banbury Road, Oxford, OX2 6NN.

Howard Noble, howard.noble@oucs.ox.ac.uk
Oxford University, 13 Banbury Road, Oxford, OX2 6NN

ABSTRACT

The Modelling4All Project is building the *BehaviourComposer*, a web-based tool for constructing, running, visualising, analysing, and sharing agent-based models. These models can be constructed by non-experts by composing pre-built modular components called *micro-behaviours*. We are attempting to seed and nurture a Web 2.0 community to support modelling. Models, micro-behaviours, lesson plans, tutorials, and other supporting material can be shared, discussed, reviewed, rated, and tagged.

The *BehaviourComposer* supports a *middle-out* style of constructionist learning. Rather than begin with primitives of a programming system and build upward, learners browse for pre-built code fragments. They then customise and compose these components to quickly build rich agent-based models. For learners interested in particular scientific topics (e.g. how epidemics spread, how fish school, or how unequal wealth distributions can emerge) this provides a means to explore their scientific interests without first investing time and effort in mastering a programming language. For teachers in life, physical, or social sciences it provides a tool enabling their students to creatively learn in an exploratory fashion both the scientific topic being model as well as learning about the computer modelling more generally. Importantly the model components are transparent, in that they are small bits of readable NetLogo code that ambitious students and teachers can delve into and edit or author new ones.

The BehaviourComposer also supports *collaborative* model building in a similar manner to how Web 2.0 tools such as wikis and Google Docs support collaborative authoring. The models are hosted on the Modelling4All web site where geographically separated collaborators can make contributions to a shared model. They can enable real-time collaboration where changes any of them make are seen by the others in a few seconds. In addition to collaborating on model construction the web-based nature of the BehaviourComposer makes it easier to have collaborations where some author the models, others test or analyse the models, while others produce guides or video tutorials. Another form of collaboration is where some who are good NetLogo programmers author new specialised micro-behaviours while others without those skills explore the combinatorial possibilities of those behaviours to build a range of models.

Keywords

Agent-based modelling, NetLogo, simulation construction kits, micro-behaviours, BehaviourComposer, Web 2.0

SOCIAL SUPPORT FOR MODELLING BY NON-EXPERTS

The Modelling4All Project began by building upon the results of the Constructing2Learn Project [1, 2] also at Oxford University in which a modelling tool called the *BehaviourComposer* was designed, implemented, and deployed for use by students. The *BehaviourComposer* had a web browser component for browsing web sites of code fragments called micro-behaviours. These are bits of code that were carefully designed to be easily understood, composed, and parameterised. The *BehaviourComposer* user attached these micro-behaviours to prototype agents. In order to create models containing many instances of a prototype agent, a micro-behaviour for making copies was added to the prototype. When the user wished to run the current model, the *BehaviourComposer* assembled a complete program and launched it. The program assembled *NetLogo* [3] programs, but the framework could be adapted for other modelling systems such as *Repast* [4].

The Modelling4All Project has constructed the *BehaviourComposer 2.0* which is a complete redesign and re-implementation of *BehaviourComposer* in order to run in web browsers. There are many advantages to providing applications via web browsers. In many organisations, universities, and schools computer systems are “locked down” and only administrators can install or upgrade software. BehaviourComposer 2.0 allows users to save their work on servers, facilitating sharing and mobile use. Web browsers exist in nearly every operating system and on many mobile devices. The system is easy to use because the user interface builds upon the familiar web browser interface.

The Modelling4All Project has another reason for choosing a web-based approach. We are striving to build a web site (http://modelling4all.org) to support an online community as they design, build, analyse, validate, and verify models. We see great potential in using the Web 2.0 concepts that have been so successful in sites such as Wikipedia, flickr, YouTube, del.icio.us, and FaceBook. We have designed *BehaviourComposer 2.0* to facilitate embedding it and the models users create in other web-based tools. In this way a community of modellers can share, discuss, review, rate, and categorise the models, micro-behaviours, and supporting materials. Users can embed their models in their blogs, wikis, web sites, discussion forums, and email.

CREATING MODELS BY COMPOSING MICRO-BEHAVIOURS

BehaviourComposer 2.0 provides libraries of generic micro-behaviours organized into categories for specifying the initial state of agents, movement, appearance, attribute maintenance, scheduling, layout, copying, interactions, links, and social networks. In addition there are micro-behaviours for creating graphs, histograms, sliders, buttons, and event logs. Specialised libraries of micro-behaviours have been created for modelling epidemics, collective decision making, network formation, predator/prey ecologies, artificial economies, and low carbon ICT.

These libraries of micro-behaviours have been created by the Modelling4All team, but *BehaviourComposer 2.0* can use micro-behaviours hosted on any web site. A micro-behaviour can be authored by any web page creation software (including wikis). The *BehaviourComposer 2.0* processes the micro-behaviour web pages to add buttons to facilitate using or editing the micro-behaviour.

Users construct models in *BehaviourComposer 2.0* by adding micro-behaviours to prototypical agents. They can focus initially on getting a single individual of each “type” to behave correctly.

Then they can add a micro-behaviour to create the desired number of copies of the prototype. The fresh copies can easily be given additional behaviours to produce a heterogeneous population.

Micro-behaviours should not be confused with the software engineering concept of modules, components, or other programming language abstractions such as packages, classes, methods, or procedures. These modular constructs have interfaces that must be carefully matched in order to combine them. They represent program fragments that run only if another fragment invokes them. In contrast, micro-behaviours run as independent processes, threads or repeatedly scheduled events. They are designed to run simultaneously with a minimum (and in most cases zero) need to coordinate their execution order and interactions. Micro-behaviours resemble the structured processes in the *LO* programming language [5].

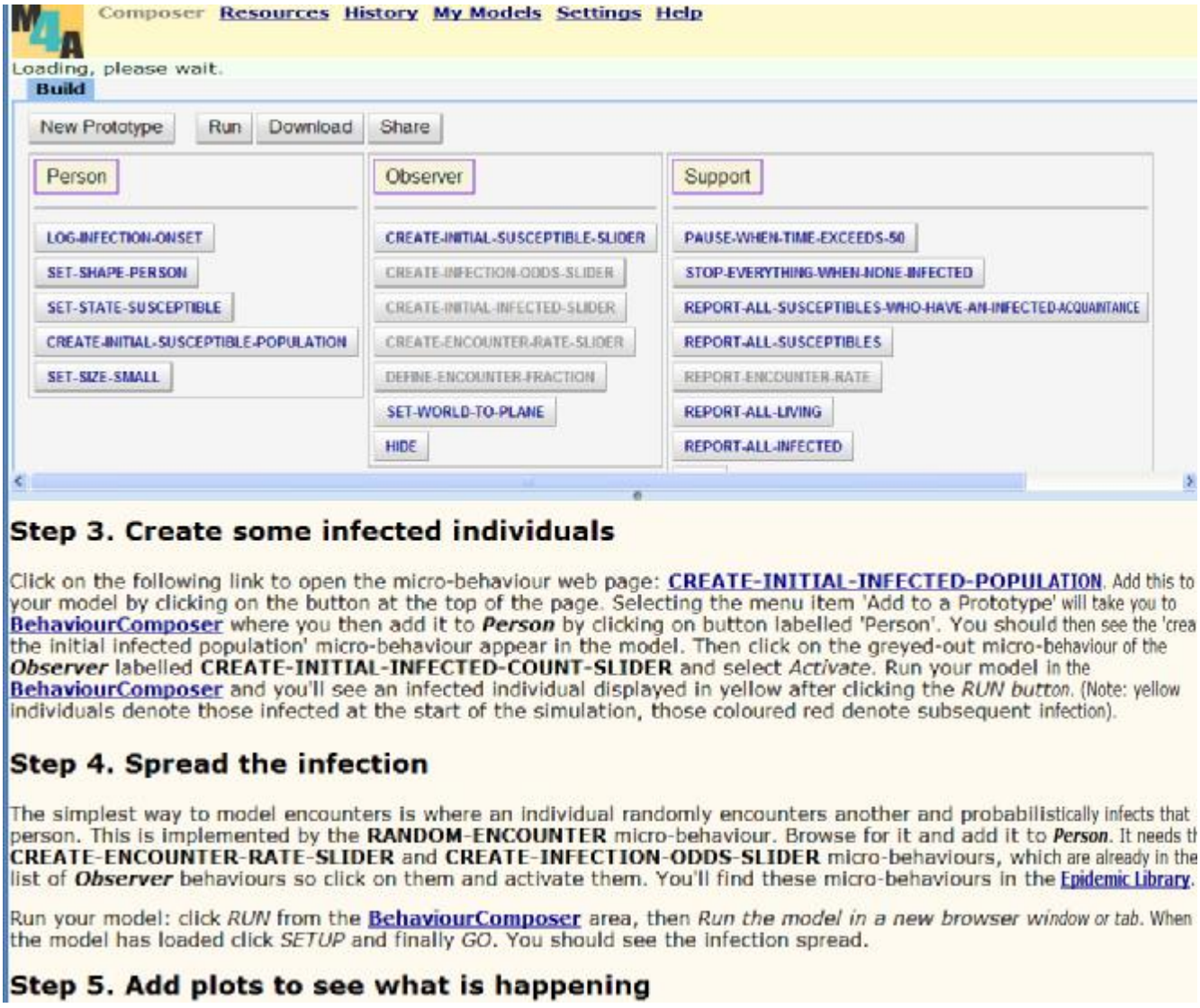

Figure1. Screen shot while constructing a model following a guide (in the bottom pane)

An illustrative example of a micro-behaviour is identified by the URL: http://modelling4all.nsms.ox.ac.uk/Resources/Composer/en/MB.4/RANDOM-ENCOUNTER.html. It contains the following code fragment:

```
do-every 1
   [do-if my-state = "infected"
       [do-for-n
           the-encounter-rate
           all-individuals with [my-state != "dead"]
           [set my-last-encounter the-other
            add-behaviours list-of-micro-behaviours "Encounter Behaviours"
                           [POSSIBLE-INFECTION.html]]]]
```

Our *NetLogo* extension `do-every` is critical for composing micro-behaviours. It repeatedly schedules an action that conditionally adds the POSSIBLE-INFECTION micro-behaviour. The reliance upon a scheduler associated with each agent greatly facilitates the composition of micro-behaviours without concern for component interfaces. This code fragment references another micro-behaviour POSSIBLE-INFECTION by providing the URL hosting the micro-behaviour. One source of name conflicts resulting from composing components is avoided by using the World Wide Web's global name space of URLs.

AUTHORING, CUSTOMISING, GROUPING, AND SHARING MICRO-BEHAVIOURS

It is relatively easy for a programmer to create micro-behaviours for specific uses. There are many challenges in creating *reusable* micro-behaviours including:

- Enabling non-programmers to easily specify parameters
- Enabling non-programmers to easily specify references from a micro-behaviour to other micro-behaviours
- Sharing attributes between micro-behaviours
- Properly scheduling micro-behaviours that depend upon other micro-behaviours

The first two challenges are largely met by the user interface of the BehaviourComposer. Web browsers support text areas where users can enter text (typically as part of the process of filling out forms). These text areas are used to provide easily editable parameter values (the '9' in Figure 2, for example). A micro-behaviour that references lists of other micro-behaviours can use the same interface that is used for collecting a list of micro-behaviours for prototype agents (see Figures 1 and 3).

There is no software support for dealing with conflicts between micro-behaviours that use the same attribute name for different purposes. Programmers need to trade-off between short simple names with their ease of reading, writing, and the clarity of references (e.g. in the history tab) with long names that perhaps even include the names of authors or projects involved. The default library of micro-behaviours uses names such as 'my-state', 'my-age', and 'my-acquaintances'. Perhaps as the community of micro-behaviour authors grows we will need to reconsider these names and use names such as 'my-flu-infection-state' instead of 'my-state'.

Early versions of the BehaviourComposer required that micro-behaviours be authored in HTML. HTML tags and attributes were used to identify the name and code fragment on a micro-behaviour page. References from a micro-behaviour to other micro-behaviours relied upon web page links. Editable parameters relied upon HTML *TextArea* elements embedded on the page. In addition to these essential uses of HTML, it is heavily used for rich text formatting and

providing links to related or background materials. While anyone capable of constructing computer programs can easily master HTML we discovered this reliance of HTML authoring limited hosting possibilities for micro-behaviour pages.

Due to the popularity of tools such as wikis, blogs, virtual learning environments, and other online web page creation tools, we needed to provide plain text alternatives to HTML authoring. The Modelling4All server code was enhanced to accept micro-behaviour pages with special textual tokens and transform these pages to the necessary HTML. These pages can of course still use HTML for rich text and links and all the online authoring tools support this. Few online web page authoring tools support the ability to add text areas or attributes to elements. As example of avoiding reliance upon HTML consider the micro-behaviour for creating copies of an agent (). It contains this HTML fragment:

```
<font size="2" color="gray">Begin micro-behaviour</font>
<p><b>ADD-COPIES</b></p>
<font size="2" color="gray">Begin NetLogo code:</font>
<pre>
substitute-text-area-for number-of-copies 9
add-copies number-of-copies
           list-of-micro-behaviours "Additional behaviours"
                                    [SET-RANDOM-POSITION.html SET-RANDOM-
HEADING.html]
</pre>
<font size="2" color="gray">End NetLogo code</font>
```

The HTML elements are serving minor roles. When stripped of all HTML it still functions properly. Here is a plain text version (where the special tokens are depicted in bold face):

```
Begin micro-behaviour
ADD-COPIES
Begin NetLogo code:
substitute-text-area-for number-of-copies 9
add-copies number-of-copies
           list-of-micro-behaviours "Additional behaviours"
                                    [SET-RANDOM-POSITION.html
SET-RANDOM-HEADING.html]
End NetLogo code
```

The server proceeds by first extracting the name of the micro-behaviour ‘ADD-COPIES’ and the code fragment that calls the NetLogo procedure ‘add-copies’. The code is transformed to replace ‘number-of-copies’ with an HTML text area initially containing ‘9’. The list of micro-behaviours is replaced by a custom GWT widget that provides an editable list of “live” micro-behaviours. The resulting HTML is rendered as in Figure 3 (typically there is surrounding rich text providing documentation, variants, related behaviours, etc.).

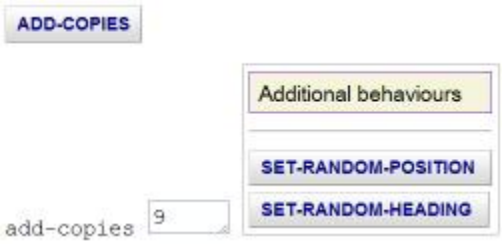

Figure 2. Screen shot of the essential part of the ADD-COPIES micro-behaviour

The only difference a user would see if the source page was stripped of all HTML would be that the name '`ADD-COPIES`' would not be in bold face. The ability to use arbitrary HTML in the name of micro-behaviours does provide a way to augment or replace names with icons or images.

Another problem with using wikis or blogs to host web pages is the hosting program adds additional material (navigation aids, help buttons, editor controls, and sometimes advertisements). We enable authors to select just a portion of a page to appear within the BehaviourComposer by adding special start and end tokens.

An additional benefit of defining micro-behaviours using unique tokens to identify elements is that search engines can find micro-behaviours using search terms such as "Begin micro-behaviour". If the search is carried out within the BehaviourComposer then the micro-behaviour pages are presented in their processed form. Any search engine can be queried from within the BehaviourComposer.

A WEB-BASED MODELLING TOOL

BehaviourComposer 2.0 is built upon the *Google Web Toolkit* (GWT) [6] and *NetLogo*. *BehaviourComposer 2.0* is a rich internet application (a web application with features comparable to desktop applications) using AJAX [7]. GWT supports interface elements such as tabs, panels, buttons, and editors as well as facilitating communication with servers. Users interactively assemble micro-behaviours into collections that represent prototypical agents. When the user clicks the *run* button, the server assembles a complete *NetLogo* program. The user can then run the program in their browser as a Java applet or download the program into *NetLogo*.

BehaviourComposer 2.0 supports micro-behaviours that use *NetLogo's* facilities for animating simulations, providing sliders for interactively exploring the parameter space, producing dynamical graphs, and interactively running experiments. Other *NetLogo* tools such as the *BehaviorSpace* for automating the exploration of the parameter space and gathering statistics are only available after launching *NetLogo* as an application rather than a browser applet.

Each micro-behaviour is presented as a web page which can be accessed via links, tags, or a search engine just like any other web page. Browsing for micro-behaviours uses the same tools and skills as web browsing for any other kind of information. New tools and skills do not have to be mastered.

A section of the web page is the program fragment itself (see Figure 3). A button is automatically generated when the page is loaded into *BehaviourComposer 2.0*. When the button is pushed, the code fragment is added to the desired prototype agent or list of micro-behaviours. By convention, the rest of the page includes sections that

- describe the behaviour
- describe how to edit the micro-behaviour to produce variants

- provide links to related micro-behaviours
- describe how the program fragment implements the desired behaviour
- a history of edits to the micro-behaviour

Some pages also have references to published papers and links to sample models using the behaviour. The addition of formal specifications of micro-behaviours is a topic of future research.

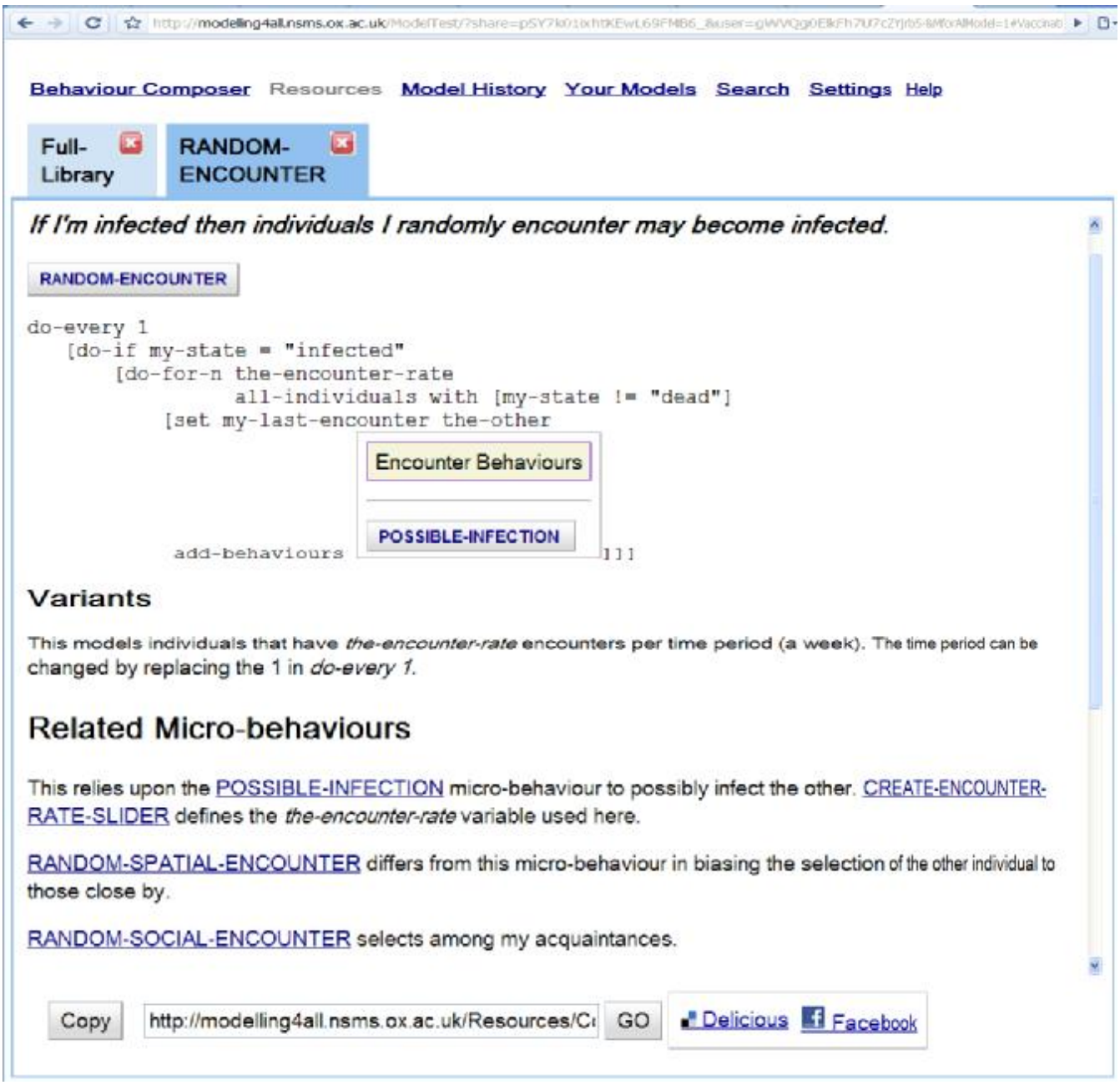

Figure 3. Screen shot of a micro-behaviour

The identity of a micro-behaviour is its URL. A micro-behaviour that references other micro-behaviours (e.g. adding new micro-behaviours to other agents) does so by providing web links to the referenced micro-behaviours. The "owner" of the URL can then update the contents to upgrade the micro-behaviour. Model makers, who instead want a snapshot of the current micro-behaviour, need to copy the contents of the page to another URL. The Modelling4All web site

supports this copying (or editing) of micro-behaviours and produces new URLs with unique identifiers.

By relying upon URLs we avoid any possibility of name conflicts between micro-behaviours. It does introduce reliance upon the "good behaviour" of micro-behaviour web page authors. It also potentially introduces additional points of failure if a model relies upon web pages that are not currently available. The Modelling4All server does cache the code fragments of micro-behaviours but this introduces new issues of staleness of items in the cache. We are exploring policies that inform users when either the cached code differs from the current or the web page is inaccessible.

We have recently been exploring a style of usage of the BehaviourComposer that builds models from very generic building blocks. For example the RANDOM-ENCOUNTER behaviour in Figure 3 could alternatively be built by adding a DO-IF behaviour to the DO-EVERY behaviour. A DO-FOR-N behaviour could then be added to the DO-IF. UPDATE-ATTRIBUTE and ADD-BEHAVIOURS micro-behaviours could be added to the DO-FOR-N behaviour. These micro-behaviours are given more appropriate names and parameterised appropriately. We have found that most specialised micro-behaviours can be replaced by a composition of appropriate generic micro-behaviours in this manner.

Every change made to the model (adding or removing micro-behaviours, adding or removing prototypes, renaming, updating parameters, or loading sub-models) is communicated to the server. The server maintains a session history that is identified with a globally unique identifier. A user can resume a session either by relying upon their browser's cookie mechanism or by bookmarking a session URL. Small teams can share a session ID to facilitate collaboration. They can choose a real-time collaboration option so that changes made to the model are seen within a few seconds by all sharing the session.

Sessions are integrated with the browser's history facility. Any changes to a model can be undone by using the browser's *back* button. They can then be restored using the browser's *forward* button. *BehaviourComposer 2.0* has a history tab that lists descriptions of every model change. By clicking on entries users can restore the model to any point in its history.

When a user runs a model they are presented with new tabs that enable the user to execute it, embedded it in various ways in other web pages, or to export their model as XML. Models can be shared with others in several ways:

- as a *snapshot* that enables others to create a copy of the model at the time it was created and make changes to their copy
- as a *locked model* that enables others to create a copy of the latest version of the model and make changes to their copy
- as an *unlocked model* that enables users to access and make shared updates of the latest version of the model (all users of an unlocked model can roll back to earlier versions)

One advantage of providing a web-based model authoring tool is that the user is relieved of file and version management. Users need not concern themselves with transferring files in order to continue working on a different computer. Files are backed up automatically. Unlike desktop applications, users need not think about different versions of file formats since the server can automatically update internal files or databases. Giving others the opportunity to run, copy, or modify one's models is accomplished by sending them the appropriate URLs.

Another advantage of running our modelling tool within a browser is that it enables a tighter integration of associated resources. Libraries of micro-behaviours, tutorials, construction guides, lesson plans, and documentation can be HTML pages. *BehaviourComposer 2.0* can integrate these resources as tabs within the application. These pages can easily have "live" entities such as buttons for micro-behaviours or adding models and sub-models. It is particularly convenient

to simultaneously read and access resources and build a model when *BehaviourComposer 2.0* is run in split screen mode. See Figure 1.

A web-based tool benefits from the tremendous world-wide efforts to improve the web and browsers. One example of this is cascading style sheets (CSS). CSS is used for all the user interface elements of *BehaviourComposer 2.0*. The styles can not only be changed to suit different tastes but also used to improve usability in special contexts such as mobile devices with small screens or visually impaired users.

AS A WEB 2.0 COMPONENT

Rather than build a large monolithic model authoring web application that also supports tagging, discussions, rating, usage summaries, and custom collections of creations we designed *BehaviourComposer 2.0* to be focused upon model authoring and to integrate well with Web 2.0 services provided by third parties.

The Modelling4All web site does not publish models. Instead models are always available via URLs containing unique global unguessable identifiers. These URLs provide privacy which is often desired for work-in-progress. No models are accessible unless their URLs have been published elsewhere. They become public only after a user references their model's URL in a blog, wiki, web site, email forum, or any other place where search engine spiders can find them.

The Modelling4All web site does not require a user to register and log in. Anyone can use it including spammers, vandals, and other troublemakers. However, since the site only produces unique URLs and does not publish anything created on the site, there is little harm they can cause and little that they can gain from doing so. This relieves us of much of the need to police user generated content for porn, copyrighted material, and other illegal material.

While we don't require login we still support a kind of authorisation that relies instead upon having unforgeable unique URLs. Only someone holding a session URL, for example, can access or change that session. At the Modelling4All site permission to use resources is not based upon identity but upon having obtained unique URLs. This approach builds upon the concepts of capability-based security. [8]

There is added value in supporting a minimal notion of identity. If the site can connect the identity of different authoring sessions then users could search for any of their past work. Collaborations are easier to manage if all parties agree to use the team's user identity. The Modelling4All site supports this weak sense of user identity. We rely upon unique unforgeable identifiers to represent users. The site does not know the identity of its users but can determine if the same user (or team sharing an identity) contributed to different sessions. The identity mechanism could be enhanced to give teachers access to the work of their students while the students only have access to their own work.

We believe that hosting models, sessions, and micro-behaviour edits in a private anonymous manner facilitates the integration of our services with third party services. The Modelling4All site hosts resources but does not make those resources accessible to those lacking the appropriate unique identifiers. Only if the holders of those identifiers make them publicly available on other web sites do the resources become available to the public. One of the problems with combining different Web 2.0 services is that each service typically has its own notions of identity and authorisation. The Modelling4All site does not contribute to this problem since it treats users and resources as anonymous.

We considered directly supporting discussion threads associated with saved models and instead have demonstrated how such threads can be hosted elsewhere (e.g. GoogleGroups). They can be embedded on the same web page as a Modelling4All model. The authors of the micro-behaviours and models hence decide where their creations will be discussed. We provide

exemplars that point to the recommended practice for providing a discussion forum for micro-behaviours and models.

Social tagging has proved to be a useful way of categorising and organising large collections in a bottom-up fashion. The Modelling4All software provides buttons to add material to popular tagging sites such as del.icio.us. We are exploring stronger integrations with tagging sites using the site's APIs that would simplify the adding tags or using them for navigation. Because the tags are added to social bookmarking sites a folksonomy of micro-behaviours should emerge.

Users of Web 2.0 sites are guided by the ratings that earlier users have given to their pages. A rating facility will be added to the Modelling4All site so that users can find the highly rated models, micro-behaviours, and supporting materials.

In addition to ratings, users find it valuable to know the relative popularity of resources. We plan to add feeds that can be turned into user-friendly configurable gadgets (e.g. iGoogle gadgets) that can display lists such as models most frequently run, micro-behaviours most heavily used, and models most frequently copied and extended. These statistics will be produced in such a way that private models are not revealed.

We are investigating the possibilities of integrating forms for collecting data with model building guides. We constructed two guides which refer to forms that feed data into shared Google spreadsheets. The aggregated data is updated in real-time and can be made available in a classroom setting to the contributing students. It could also facilitate a teacher who wanted to collect the results each student obtained from running their model. The data collection, construction guide, and model authoring can all be integrated together.

AS A TEACHING TOOL

We have used the Modelling4All web site and tools in classrooms at Oxford University. We worked with instructors in producing micro-behaviour libraries tuned for modelling the desired subject matter and associated construction guides.

About 30 third year biology students constructed and ran a series of models exploring the dynamics of an epidemic spreading over a social network. In a single session they were able to build models with different kinds of networks and interventions. During the session they ran several variant models and each student contributed to a spreadsheet that automatically collected the reported results from a series of simulation runs.

Two groups of MBA and MSc students at the Oxford University Said Business School constructed and ran a series of Sugarscape models [9]. In a two-hour session most were able to build the models described in chapter two of the book *Growing Artificial Societies: Social Science from the Bottom Up*.

We have run a workshop with Oxford University academics and students where they built a predator prey model.

We have scheduled a session with economics students where they will use the site to build a series of models exploring network formation.

Very few of the biology or business school students had any computer programming experience, and yet they were able to build serious models in their field of study. They learned about the behaviour of a complex system in their subject as well as acquiring some understanding of the general process of model construction. They acquired what one of the faculty members we worked with calls *modelling literacy* – an understanding of how simulations work and the ways in which they are designed and constructed.

The students who built models of epidemics had an earlier session where they built a simple mathematical model of epidemics using other software. This modelled the dynamics of entire populations. When using the Modelling4All site they began with an agent-based model that mirrored this aggregate model. They then went on to explore the consequences of modelling a heterogeneous population. Agent-based models produce different dynamics of epidemics and outcomes for interventions than aggregate models do.

Another learning outcome is an appreciation for the differences between emergent phenomena and top-down control. The business school students, for example, saw how even very simple bottom-up models produced uneven wealth distributions that increased over time.

The micro-behaviours were designed to be engaging at different depths. A shallow understanding of a micro-behaviour is purely functional – what does it do and how can it be used. Some students were also concerned with how the micro-behaviours work and how they could be modified. The micro-behaviours by convention have a section explaining how they work. Additionally a good deal of effort went into making the source code readable by non-experts. In this way, the Modelling4All classroom session could be a first step towards learning to computer programming for building models.

AS A PUBLIC ENGAGEMENT IN SCIENCE TOOL

We are building a specialised version of the BehaviourComposer for the Royal Society 2010 Summer Science Exhibition. The challenge is to give visitors a taste of constructionist learning in only a few minutes. We will present visitors with the Epidemic Game Maker that contains a very simple model of an epidemic and a single intervention that players can take to stop the epidemic. Rather than expect the visitors to have the time and patience of browsing through a library of micro-behaviours for enhancing both the model and the game play we are providing customised buttons that enhance the game. Visitors can choose what entities to add to the model, what new behaviours, what things to measure, etc. by choosing which buttons to click.

ANIMATING MODEL EXECUTION IN *SECOND LIFE* AND MAPPING SERVICES

We have implemented a way to see the execution of a model inside of the 3D online virtual world *Second Life*. We provide a URL for each model that produces a stream of values for the position, orientation, scale, and colour of each agent in the model. The stream also records the creation or destruction of agents. Scripts within *Second Life* repeatedly read this stream and recreate the trace with *Second Life* objects representing the *NetLogo* agents. This is accomplished by running the model on our servers. By doing this model executions can be experienced in a social immersive manner.

We have also implemented a primitive manner for commands to be sent from within *Second Life* to alter a running model on our servers. We plan to add *Second Life* interface objects that play the same role as sliders and buttons within *NetLogo*. These interface objects will work by sending commands to the model running in *NetLogo* on our servers.

We have begun work on taking this same approach to visualising the execution of our models on mapping services such as *Google Earth* or *Google Maps*. The Modelling4All server code will be enhanced to produce standard mapping files [10] that can be loaded into a variety of mapping services. We plan, for example, to animate the execution of a model of pandemics on *Google Earth* enabling the viewer to see the spread of the virus on a globe they can view from any angle or height.

POTENTIAL PROBLEMS

One problem with providing a tool as a service is that users rely upon the service provider to maintain a robust stable service. This is a relatively minor problem when a large corporation such as Google or Microsoft provides the service but when it is provided by a small team in university project there is a greater concern. The problem is alleviated somewhat by providing a way to export one's data, by releasing the source code for the system, and by providing a way to copy models between different servers running the Modelling4All code.

Another problem is that the system is currently impossible to use without an Internet connection. Serious users can run a Modelling4All server locally to overcome this. A promising alternative we are considering is to integrate *Google Gears* [11] with *BehaviourComposer 2.0*. *Google Gears* is a browser plug-in that provides local storage. Using *Google Gears* the software could continue to work in some cases without a network connection, and then models will be uploaded when the connection is established.

We may discover difficulties with version management, especially for micro-behaviours. Software developers typically rely upon version control systems so that each build relies upon the appropriate version of components. In the *BehaviourComposer 2.0* references to micro-behaviours are by fixed URLs. The parties hosting those micro-behaviours can change the contents of the web pages, perhaps breaking models that relied upon the old contents. In contrast, Modelling4All models can be flexibly shared as a frozen version, as a read-only copy of the latest version, and as read-write access to the latest version. Perhaps we will discover that micro-behaviours need similar version control. It is possible to build web sites for micro-behaviours where the URLs specify the desired version policy.

Some are concerned about the public nature of HTTP traffic between the model maker and the servers. This could be alleviated by using a secure connection (HTTPS). Passwords could be automatically provided since here we are only trying to encrypt communications with the site for privacy reasons.

Models can be created and edited without the use of browser plug-ins. Many potential users with browsers lacking a plug-in will not, or are not allowed to, install a plug-in. Because BehaviourComposer 2.0 currently only supports *NetLogo*, the running of models requires either a plug-in to run Java applets or the prior installation of NetLogo (free for educational and research purposes). While the plug-in for Java applets is installed in the majority of browsers this remains a problem for a large minority of users. A sister project to Modelling4All has developed MoPiX [12] where the execution and animation of models is performed by the browser without the need for any plug-ins. BehaviourComposer 2.0 is, however, much more expressive than the equational programming supported by MoPiX.

There is concern that by giving users the freedom to choose the Web 2.0 tools that they integrate with their use the Modelling4All web site that the community will be much more fragmented than if a monolithic Web 2.0 site were provided instead. By providing guidance and exemplars we hope to guide community members towards shared tools.

The general issue underlying web applications is loss of control [13]. Students in a university or school have typically already lost control of the computers and software they use. This is more of an issue for long-term research projects using our services. Since Modelling4All is an open-source project, full control can be obtained by running the server locally. This might, however, fragment the community.

POTENTIAL USES AND FURTHER DEVELOPMENTS

The Modelling4All software currently only supports micro-behaviours constructed in *NetLogo*. The idea of browsing for program fragments that can be combined using a web browser can be applied to other modelling tools. If the language supports the expression of modular micro-behaviours then the server can generate complete source files that can be compiled and executed.

As users construct and run models, our servers accumulate data about how the site is being used. This data could be mined to focus further development efforts on those aspects that are most crucial. For example, analysis of the usage data may discover a common stumbling block where a significant fraction of users get stuck. We can then work to address this problem. Or we may discover that a very useful and powerful facility is being overlooked and we can then promote its use.

Data mining could also be useful to the Modelling4All community to acquire a crude level of self-awareness. Community members could learn what others are doing. Teachers could obtain summaries of what their students have built on the site.

To date we have focussed upon the educational uses of the Modelling4All site. We believe that researchers, journalists, and, policy makers could profitably use the site. It could also be valuable to the general public attempting to understand more deeply topical subjects such as causes of global warming or the spread of HIV. Some visitors to the site may only run a few highlighted models while some may follow the tutorials and construction guides to obtain a much deeper understanding of the underlying processes and mechanisms. We hope that our efforts to build and test the Epidemic Game Maker will address these concerns.

ACKNOWLEDGEMENTS

We are grateful to the Eduserv Foundation who has funded and supported this research. We want to thank the Oxford University Computing Services for their continued support. The JISC funded the Constructing2Learn Project that this work builds upon.

REFERENCES

Kahn, K., *Comparing Multi-Agent Models Composed from Micro-Behaviours*, Third International Model-to-Model Workshop, Marseille, France, March 2007

Kahn, K. *Building Computer Models from Small Pieces*, 2007 Summer Computer Simulation Conference, San Diego, CA, July 2007.

Wilensky, U., *NetLogo*, Center for Connected Learning and Computer-Based Modeling, Northwestern University, http://ccl.northwestern.edu/NetLogo/

North, M.J., Collier, N.T. and Vos, J. R., *Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit*' ACM Transactions on Modeling and Computer Simulation, Vol. 16, Issue 1, pp. 1-25, ACM, New York.

Andreoli, J. and Pareschi, R., *LO and behold!* Concurrent structured processes, Proceedings of the European Conference on Object-Oriented Programming on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA/ECOOP '90, Ottawa, Canada, ACM Press. Also published in ACM SIGPLAN Notices, Volume 25, Issue 10, Oct. 1990.

Google Web Toolkit, http://code.google.com/webtoolkit/

AJAX, http://en.wikipedia.org/wiki/AJAX

Capability-based security, http://en.wikipedia.org/wiki/Capabilities

Epstein, J. and Axtell, R., *Growing Artificial Societies: Social Science from the Bottom Up*, Brookings Institution Press and MIT Press, 1996

Keyhole Markup Language, http://en.wikipedia.org/wiki/Keyhole_Markup_Language

Google Gears, http://gears.google.com/

MoPiX, http://www.lkl.ac.uk/mopix/

Johnson, B. *Cloud computing is a trap, warns GNU founder Richard Stallman*, http://www.guardian.co.uk/technology/2008/sep/29/cloud.computing.richard.stallman, Monday September 29 2008 14.11 BST