The Modelling4All Project
A web-based modelling tool embedded in Web 2.0

Ken Kahn
Oxford University
13 Banbury Road
Oxford, OX2 6NN
+44 1865 283 377
Kenneth.Kahn@oucs.ox.ac.uk

Howard Noble
Oxford University
13 Banbury Road
Oxford, OX2 6NN
+44 1865 273 211
Howard.Noble@oucs.ox.ac.uk

ABSTRACT
The Modelling4All Project is building 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.

Categories and Subject Descriptors
I.6.2 Simulation Languages: *Modeling methodologies, agent-based, visual, discrete*. **H.3.5 Online Information Services**: *Web-based services*

General Terms
Design, Languages

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

1. SOCIAL SUPPORT FOR MODELLING BY NON-EXPERTS
The Modelling4All Project began by building upon the results of 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].

SIMUTools 2009, Rome, Italy.

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.

2. 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, reproduction, death, 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 HTML micro-behaviour web pages to add buttons to facilitate using or editing the micro-behaviour.

A major technical challenge is to design and build micro-behaviours so that they can be combined without concern for their order of execution. Each micro-behaviour is modelled as an autonomous process. A fish in a school, for example, may be concurrently running processes for avoiding fish that are too close, for aligning its orientation with neighbouring fish, for staying close to neighbouring fish, and for heading in a desired direction, as well as processes for modelling noise. These processes combine to generate the desired agent behaviour. Conflicts between these processes are avoided by careful use of scheduling routines and support for simultaneous updating of attributes (we added both to *NetLogo*).

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].

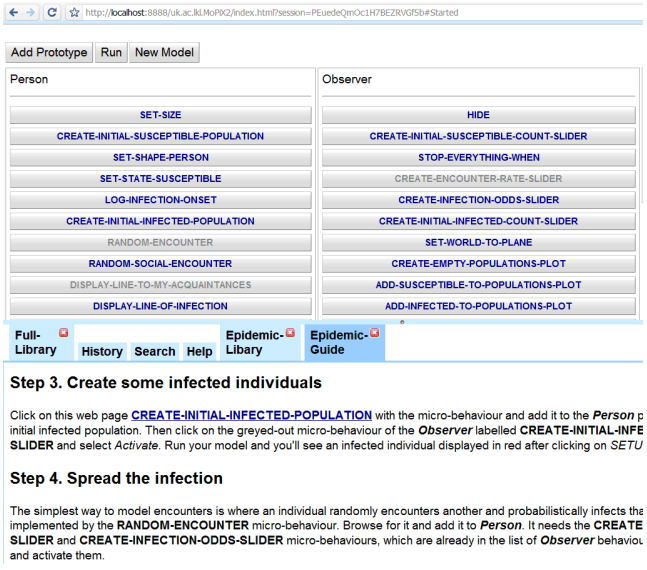

Figure 1. Screen shot while constructing a model

An illustrative example of a micro-behaviour is identified by the URL http://modelling4all.nsms.ox.ac.uk/Resources/Composer/en/MB/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-behaviour
                POSSIBLE-INFECTION]]]
```

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 a link to 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.

3. 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 an 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. 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. 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.

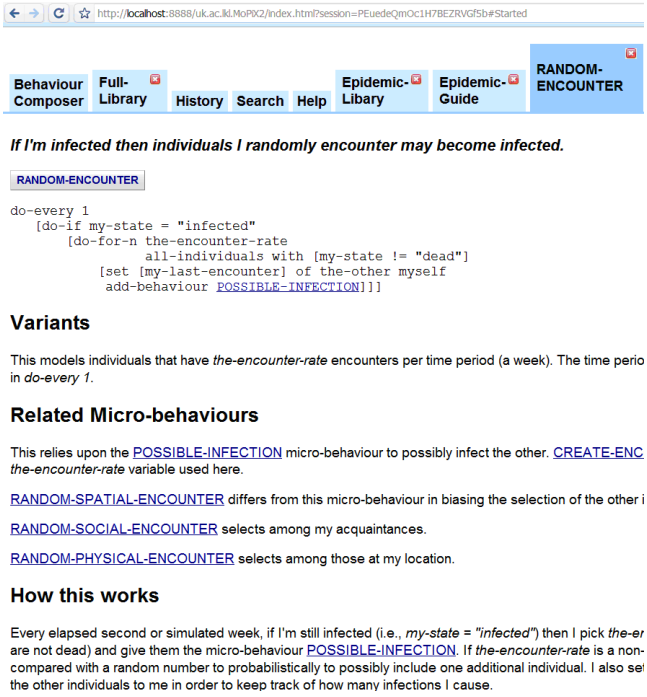

Figure 2. Screen shot of a micro-behaviour

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.

Every change made to the model (adding or removing micro-behaviours, adding or removing prototypes, renaming, 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 public updates of the latest version of the model (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. 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.

4. 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 be 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 identifers 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 by 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 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 a guide which refers to a form that feeds data into a Google spreadsheet. This could facilitate, for example, 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.

5. 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.

A group 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 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.

6. 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 and by releasing the source code for the system.

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* [10] 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 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*, running 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 is developing MoPiX [11] 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 [12]. 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 would, however, probably fragment the community.

7. 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.

We plan to explore this to produce scripts that can run the multi-user virtual environment *Second Life*. Models can be produced in *BehaviourComposer 2.0* and then imported into *Second Life*. In this way model executions can be experienced in a social immersive manner. We expect that parallel micro-behaviours can be built in both *NetLogo* and a *Second Life* scripting language so that one can build and test models in *NetLogo* and then export the parallel versions to *Second Life*.

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.

8. 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.

9. REFERENCES

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

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

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

[4] 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.

[5] 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.

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

[7] AJAX, http://en.wikipedia.org/wiki/AJAX

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

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

[10] Google Gears, http://gears.google.com/

[11] MoPiX, http://www.lkl.ac.uk/mopix/

[12] 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