Cooperative Research Centre for National Plant Biosecurity

Final Report

CRC10073

Surveillance Simulation Platform

Authors

Mark Reynolds

Juan Jose García Adeva

16 December 2011

Project Leader contact details:

Name: Mark Reynolds
Address: CSSE, UWA M002, 35 Stirling Highway, Crawley WA 6009
Phone: 08 6488 2279
Fax: 08 6488 1089
Email: mark.reynolds@uwa.edu.au

CRCNPB contact details:

Cooperative Research Centre for National Plant Biosecurity
LPO Box 5012
Bruce ACT 5012

Phone: +61 (0)2 6201 2882
Fax: +61 (0)2 6201 5067
Email: info@crcplantbiosecurity.com.au
Web: www.crcplantbiosecurity.com.au

Table of contents

1. Executive Summary

Biosecurity plays a key role in the economic viability of the plant-based industries of countries like Australia. Plant pests and diseases can have devastating effects on food safety, trade, market access, market development and, ultimately, the profitability and sustainability of plant industries. Incursion and outbreak management are key aspects to biosecurity. As soon as an incursion of plant pest or disease is detected and reported to the appropriate authorities, biosecurity managers have to apply control mechanisms in order to avoid it from spreading. Currently, researchers provide biosecurity managers with biological parameters about the pest or disease on host range, mode of spread, and potential distribution based on climatic factors. However, strategies to test pre-emptive surveillance and estimate rates of spread in a spatial environment in real time are not available.

In CRC10073, we have developed a 'what if' scenario simulation tool to address this shortcoming.

The project milestones haven successfully achieved: a generic simulator framework is in operation, specific versions for a range of particular example pests and diseases have been demonstrated, validation against actual incursion data has been piloted, details of design and validation have been published and a robust and useful graphical interface has been successfully demonstrated to potential users from DAFWA and DAFF (Federal). The current production of the project consists of a large pool of technological artefacts including computer infrastructure, support services, three simulation prototypes (i.e. Fruit fly, Fire blight, and Gypsy moth), and a generic software application framework.

These outcomes concur with the original project objectives, by offering the potential to analyse and map the habitat and incursion and also the speed and spread of the invasive pest, thus providing a valuable tool for the industry and also saving money and time (timing is one factor that is extremely important for implementation of an eradication program).

The project members have published seven conference publications, five press releases, and six journal papers have been published, are currently under review or are in the process of being submitted.

2. Aims and objectives

Aim: to produce a simulation platform to estimate rates and patterns of spread of plant diseases and pests and the time-changing extent over the landscape. It has to provide plant pest outbreak decision makers with suitable computer tools to support their decision process.

Objectives:

- To develop a generic simulation model and toolset to predict spread of Emergency Plant Pest (EPP) high-risk species.
- To use this to identify high-risk locations for various EPPs.
- To validate the technology and develop methodologies for its effective usage via replication of well-documented, historic outbreaks.

- Will provide Biosecurity scientists and managers with a real-time interactive surveillance risk analysis tool.

The main outcomes of the project are:

- a surveillance prediction simulation platform for validating surveillance strategies
- novel landscape-level modelling techniques for pest spread simulation, and
- validated simulation technology using historical emergency plant pest incursion data.

Beneficiaries:

The primary end-users of the tool are pest/disease outbreak managers. The convenient user-friendly, robust and flexible interface to a real-time simulator means that managers can quickly try a range of hypothetical containment strategies and see the likely effects on the outbreak.

The completely portable web-based access means that the calculations are carried out on the powerful servers rather than the users' computers. They will need no special hardware or software. The web-server based technology also means that extensions, upgrades and maintenance of the software can be carried out without any need for access to or interference with the computing devices of any of the end-users.

Other end-users include Biosecurity scientists and researchers needing general, flexible and robust modelling tools and frameworks. The system allows important parameters governing the spread of particular pests to be easily adjusted, if desired, depending on the latest advice of local experts.

The general framework of the simulator is designed to allow rapid 'personalisation' to novel pests and diseases. The modular re-usable design and use of the popular Java programming language and well-documented implementation means upgrades to allow modelling of new pests can be quickly installed.

These outcomes concur with the original project objectives, by offering the potential to analyse and map the habitat and incursion and also the speed and spread of the invasive pest, thus providing a valuable tool for the industry and also saving money and time (timing is one factor that is extremely important for implementation of an eradication program).

Finally, the project members have published seven conference publications, five press releases, and had six journal papers accepted, currently under revision or in the process of being submitted. These contain a range of lessons and evidence for developers of other simulation software. The lessons include entomology, mathematical modelling and software engineering aspects. The applications range from plant pest modelling, to animal biosecurity, ecology and human health.

3. Key findings

The main outcomes of the project are:

- A surveillance prediction simulation platform for validating surveillance strategies.
- Novel landscape-level modelling techniques for pest spread simulation.
- Validated simulation technology using historical emergency plant pest incursion data.

The outputs are thus not primarily 'findings', but the development of tools and techniques. These are described in detail in the main publications of the project (see list later) as well as in technical reports and our wikis (access granted by request to authors).

Here we will, using material from the publications, summarize the main features of the tools and techniques under the following headings:

- Pests covered
- Generic Simulator Framework
- Modelling Modules
- Interface and Main Functionality
- Development Environment
- Technology
- Evaluation
- Validation

An account of the details of our decisions and experiments in designing, implementing and testing the software is thus presented as our 'findings'. This account will be useful for those using the tools and those building on them in follow-on work.

3.1. Pests Covered

The simulator is engineered to be able to be conveniently modified to work for a wide variety of EPPs, but as part of the process of development and in order to have concrete demonstrations of the technology, we have released specific versions for the following three EPPs of major concern:

- Queensland Fruit Fly (*bactrocera tryoni*)
- Asian Gypsy Moth (*Lymantria dispar*), and
- Fire blight (*Erwinia amylovora*).

These three simulators are accessible and conveniently usable online from our servers for interested researchers. Please contact the authors for access and instructions.

3.2. Generic Simulator Framework

Because we aimed at providing a framework allowing fast and convenient production of simulators for other insects and other EPPs in the future, the code base is modularized and organised in appropriate layers to facilitate their reuse, modification and/or replacement.

Some modules, such as those supporting the spatial model, differ very little among different simulators. Other modules, such as those managing user interfaces, are deliberately very similar across EPPs but need certain differences. Yet other modules are completely different and may even be absent from some simulators because there are differences in the mechanisms of spread across the EPPs.

Another factor for our design was that the simulator source code should allow extending certain modules such as the dispersal mechanisms to accommodate different expert opinions and expertise.

We considered that an important concept, related to software modelling techniques, that could facilitate the practice of reusability in this work is design patterns. A design pattern `is a common solution to a common problem in a given context' (Booch et al., 1999). They describe common analysis and design fragments in order for them to be reused (Gamma et al., 1995). Design patterns are increasingly becoming the building blocks for reusable components, hence allowing architects and designers to have more control over the quality and consistency of the final implementation (Garland and Anthony, 2003)

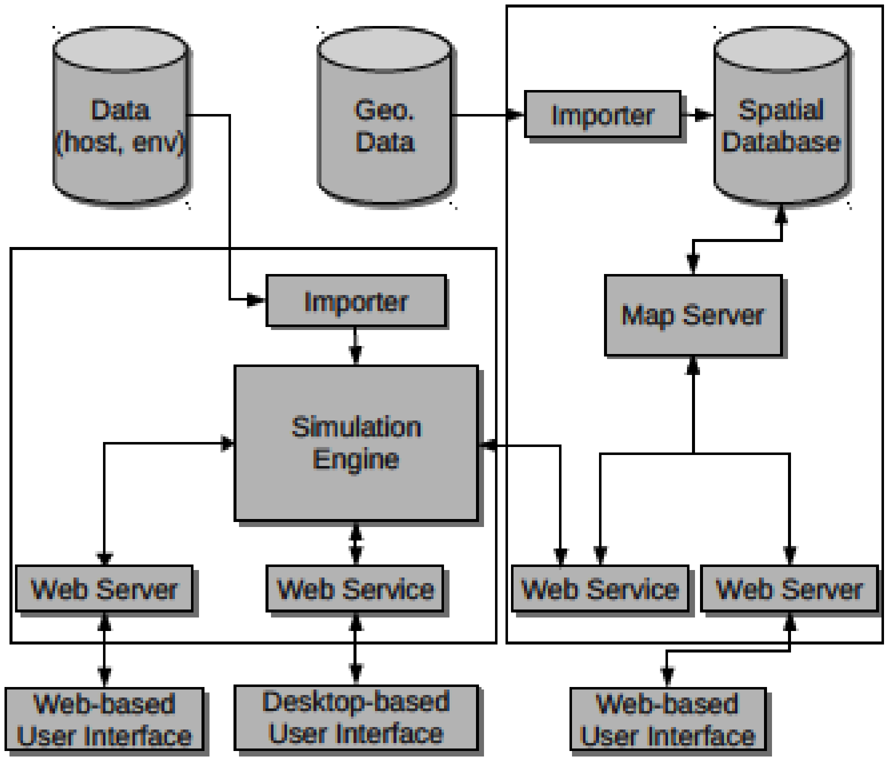


Figure 1: General Architecture of the Simulator

3.3. Modelling Modules

The simulation model implemented by the Simulation Engine includes several underlying deterministic or stochastic models that include:

- spatial model
- life-cycle model
- host seasonality model
- weather model, and
- dispersal model.

Full details of all modules are available in our publications but here is a brief description of the spatial model and the dispersal model respectively, focusing on key aspects of their implementation.

The relationship between the models is set out on Figure 2.

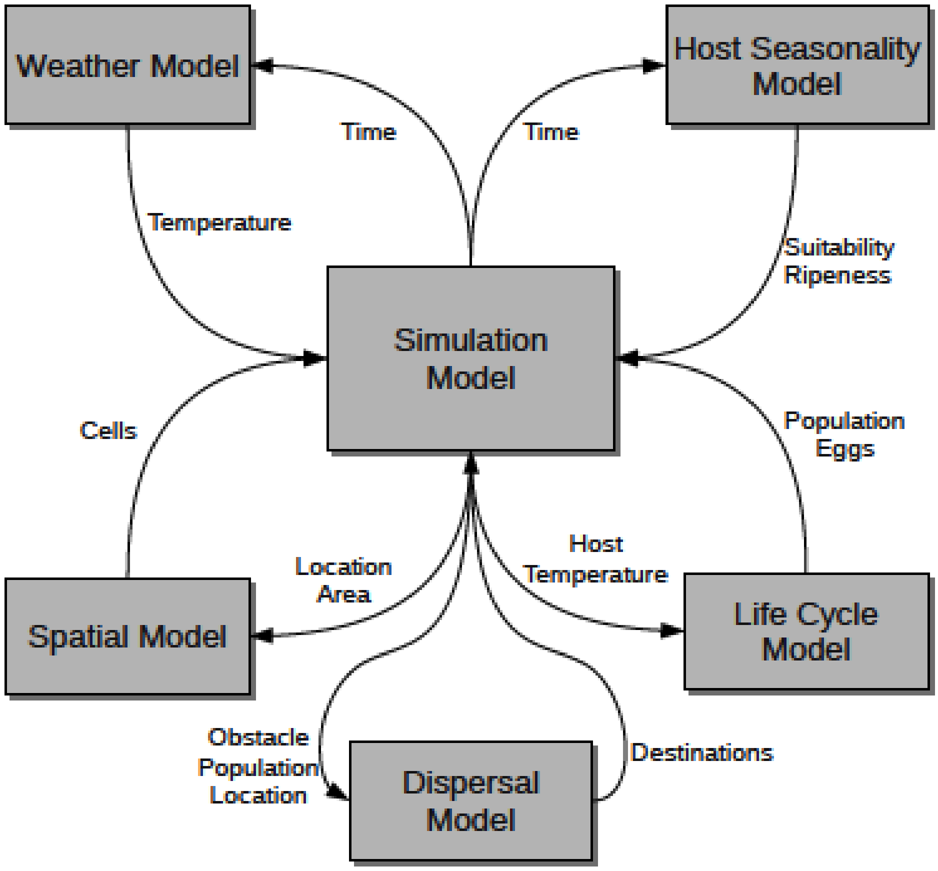


Figure 2: Conceptual diagram of the simulation model and its related models.

Spatial Model

The spatial model represents the two-dimensional simulation area and the key elements contained within it. This section describes those implementation aspects of the spatial model which are important to fulfill the requirements.

The simulation area consists of a landscape whose bounds are initially defined by the user through the simulator's user interface. Figure 3 serves as an example screenshot of the simulator when selecting a map area where to perform the simulation. The extent will usually vary from a few hectares to several hundred thousand km2 (e.g. typical simulations will happen in areas of up to 500 km by 300 km). The landscape can contain areas (i.e. polygons) representing an obstacle (e.g. a water body or a desert), illustrated by the dark area in Figure 3. Areas representing suitable habitat for fruit flies (e.g. an orchard) are depicted as shown by the two yellow areas in Figure 3. Other features such as roads, building areas, rivers, oceans, etc. are also represented. The landscape will typically contain at least one observation indicating the incursion to study, which the blue marker shown in Figure 3 represents. This landscape is represented as a rectangular planar region. The landscape boundaries, given by two global coordinates that correspond to the bottom-left and top-right corners of the extent, remain constant once the simulation area has been selected by the user. Discretisation of this landscape is performed by means of a rectangular sparse grid, where the number of horizontal cells indicates the spatial resolution and is selected by the user through the horizontal slider shown in Figure 3. The number of vertical cells and cell size are calculated based on this user-defined horizontal resolution.

The calculation used to determine the cell corresponding to a global co-ordinate applies the Spherical Law of Cosines (Banerjee, 2004) to calculate the horizontal and vertical distances from the bound bottom-left and top-right coordinates of the simulation viewing area. This is then divided by the number of cells in each dimension. The advantage of this grid configuration is that it resembles a raster image. Therefore, there is a plethora of algorithms from the area of computer graphics to take advantage of. Consequently, an area representing an orchard or an obstacle is expressed as a closed polygon defined by the list of its ordered (either clockwise or counterclockwise) vertex coordinates, each of them is converted to its corresponding cell. Frequent operations include calculating the extend of areas, achieved by using Euclidean geometry such as Meister's formula, and determining whether a particular cell is included within an area, which is accomplished by applying the Jordan curve theorem as proposed by Haines (1994).

However, the most distinctive feature of our spatial model is its sparse representation, where only those non-empty cells are created dynamically as needed. This reduces significantly initialization time, contributes to a much more flexible use of the computer memory, and improves execution time by ignoring most empty cells. Besides, those cells that become vacant (i.e. those not containing any more individuals) can be discarded when the memory resources become scarce, until they have to be created again if circumstances arise.

Dispersal Model

This stochastic model represents the ability of fruit flies to fly in order to find new habitats or resources, hence contributing to the stabilisation of their population when predation or lack of food would cause local extinctions (Goodwin et al., 2005).

From an initial population of fruit flies at a known location, the dispersal model aims to determine:

i) the speed at which this population will spread over the full extent of suitable environment available, and

ii) the distance range that this population will cover.

Therefore, given a cell in the grid containing individuals, this model determines how many of these individuals disperse and into which destinations. Dispersal is triggered by a combination of circumstances such as population density, food suitability at the individual location and surrounding areas. The model also considers dispersal barriers representing natural features such as mountain ranges, rivers, lakes, deserts, etc. Consequently, barriers will influence dispersal by limiting the spatial range of those fruit flies flying in certain directions or landing in certain areas. Also, dispersal is strongly influenced by population growth, regardless of whether this growth happens through reproduction or migration.

The dispersal distance and number of individuals dispersing at each distance is defined by a dispersal kernel, which describes the spatial distribution of propagules in the vicinity as a function of distance from the dispersing location (Garcia Adeva et al., 2011a). This kernel is based on two empirical probability distributions that allow determination of dispersal distance and direction respectively. The practical process of dispersal is based on the spatial model with the grid that represents the simulation area containing at least one cell with a number of dispersing fruit flies. For example, the blue dot in the centre of Figure 2 represents the dispersing fruit flies (i.e. propagules). The distance range between 0.5 and 2 km is represented by the black ring. The size of this ring and the probability of the propagule population dispersing within it is drawn from the empirical probability distribution representing dispersal distances as part of the dispersal kernel. All the cells included in the ring are possible destinations for propagules, with those two containing hosts in grey. The probability of these hosts being detected by the subset of propagules dispersing within this distance range is determined by the second empirical probability distribution of the dispersal kernel. Those flies that do not detect a host will choose a random direction.

The practical implementation of this process is as follows:

i) all the cells within the ring are selected in a list
ii) this list of cells is randomly shuffled
iii) those cells containing a host are moved to the head of the list if the result of applying a Bernoulli random variate (Kachitvichyanukul and Schmeiser, 1988) results in one, and
iv) propagules are distributed throughout the list.

In the case where the number of propagules is smaller than the number of destinations (i.e. size of the cell list), propagules are clustered.

3.4. Interface and Main Features

This section enumerates and briefly discusses the key features of the system produced, based on the specifications proposed, the architectural design, and our multiple implementation decisions.

- The simulation time resolution is one day. This decision was made due to the fact that the vast majority of available expert knowledge uses one day as the time step.

- The current user interface is web-based. This decision tackles especially the requirements of Availability. There are several advantages to web-based simulation over a regular local software approach (Xie and D., 2006). The simulation processes are performed in several (computer) servers hosted remotely and managed and maintained by the technical members of the project, while users access the simulator as a regular web application. This method offers many interesting advantages over the traditional desktop application approach: i) the simulator can deal with large amounts of data regardless of how powerful the user computer is, ii) no installation is required by users, the simulation models can be modified or updated without having to redistribute new copies of the software to users, and iii) access to the simulator can be controlled, adapted, or restricted if necessary.

- The simulator offers spatial features to provide users with a better understanding of the scale, extend, and location of dispersal over an actual landscape. The web-based interface includes an embedded map of Australia containing multiple selectable layers representing water bodies, roads and railways, places, suitable areas, and obstacles to spread. The simulation area and spatial resolution can be easily set by the user in order to determine a suitable balance between expected precision and time constraints during execution. Figure 3 depicts an arbitrary example of the web user interface after defining a simulation scenario located in an urban area. It shows how the embedded mapping tool includes the expected features of commercial systems such as zooming to select the area size, filtering layers, and choosing the resolution. By offering our own web mapping system, we contribute towards fulfilling the requirement of independence.

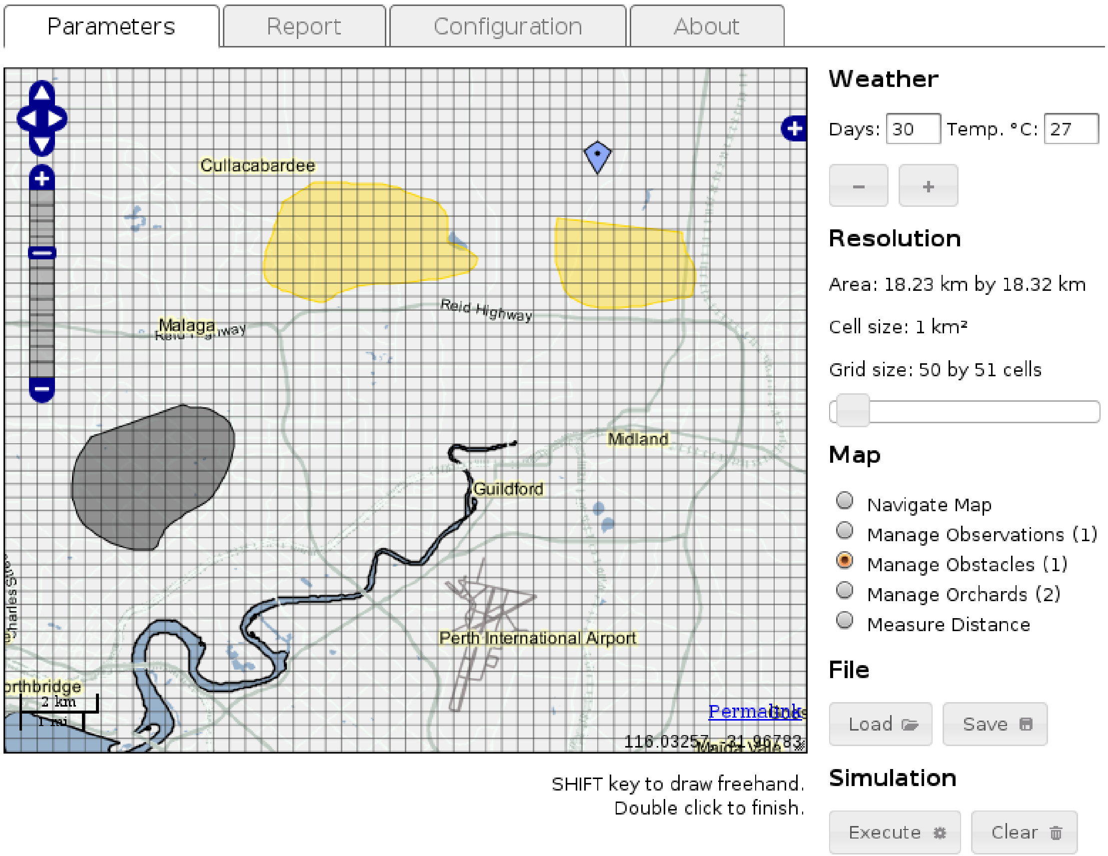


Figure 3: Example of web-based user interface in an arbitrary location in Perth (Australia) with two orchards in yellow, and obstacle in grey, and an observation.

- Performing a simulation results in an extensive report that contains all the details provided by the simulator, thus fulfilling the requirement of interactivity. The report is organised by sections, each one corresponding to one simulated day. For each day, the user can inspect: i) how weather has changed in the simulation area; ii) how the total fruit population has changed, grouped by y stages (eggs, larvae, pupae, and adults); iii) what locations have been affected by fruit y dispersal; and iv) how population has changed, grouped by stages, for each of the locations containing flies at any given time. The report also contains system information including: i) how many cells of the grid representing the simulation area have been used by day, ii) the length of time taken for the simulator to process each day, and iii) how much RAM the server used. Each of these results are provided visually graphically, and as tabular data. Moreover, they can be exported using the well-supported comma-separated values (CSV) format so that it can be easily imported into popular spreadsheet or data analysis applications.

- The simulator follows the Visual Interactive Simulation (VIS) paradigm (Hurrion, 1976; O'Keefe and Robert, 1987), implying that simulation processes occur as an animation on a visual display and the user can interact with the running model. The user can examine the simulation at any particular day and location obtaining a detailed graphical and tabular account. The user can make alterations to the model parameters and examine the new results. The benefits include i) easier verification and validation, ii) better understanding of the results, iii) more accurate communication of findings to other parties, and iv) potential to use the simulation with a group for decision making.

- It is clear that different fruit y experts sometimes differ on minor aspects of the insect behaviour; for example ranges of dispersal, oviposition, or host preferences. The underlying models can be adjusted through the user interface (i.e. the source code does not have to be altered) to reconcile multiple feedback from experts. In the current version, the model configuration is focused on settings affecting aspects of the Bactrocera fruit y life cycle such as male/fame ratio, average number of eggs laid per day, etc.

- The simulation is in real time. There are multiple definitions for what real-time simulation means. We consider real-time simulation to consist of fulfilling the conditions: i) given the same set of input parameters, multiple simulations would take approximately the same time to complete, hence providing a certain expectation with regards to time scale, and the time to complete a simulation will be equal or less than the simulated time.

- The simulator allows saving the parameters that define a simulation scenario for further reuse or sharing.

3.5. Development Environment

To design a robust and flexible piece of software that is amenable to future extension in the most convenient way, it was essential to have a supportive development environment allowing the careful control of collaboration, version control and testing.

The work environment consists of a collection of processes and software development tools, utilised and shared by all the team members, in order to work collaboratively towards a set of common goals to create the final software product, which in this case it is the simulation platform prototype.

There are several reasons why we consider a formal work environment necessary as part of the development strategy: i) it contributes to maximise programmer productivity thanks to its tools, ii) it reduces work overlap by improving communication and reuse of resources, iii) it simplifies the documentation process by providing much detailed and interrelated information during the development process.

We implemented the work environment as a three-tier scheme that includes the following tiers:

- ***Development:*** a typical development environment will comprise of a version control system (also known as revision control or source control system), a project management panel, an error tracking facility, flexible documentation tools, build cycle, and automatic documentation generation, all of them integrated. The Software Engineers working on the project will have access to all the features of this environment, while some users might have restricted access (e.g.

error tracking for users supporting the verification or validation efforts).

- ***Staging:*** it keeps the most recent version of the simulation for testing purposes, before it is deployed to the production environment. Although this tier comprises an environment that should be as identical as possible to the production environment, in cases of limited resources then it could be less powerful than the production instance. There will usually be a Software Engineer acting as Release Manager, who should take care of updating this staging environment when an appropriate number of changes have taken place to the source code base. Users such as biosecurity managers involved in the project and who are contributing to the verification or validation effort of the simulation models will use a version of the simulator hosted in this staging environment.

- ***Production:*** it keeps the latest stable version of the simulation. Most users will be provided with access to the simulator hosted in this environment. The Release Manager will be responsible for updating this environment when the simulator version in the staging environment is considered to be stable.

These tiers are referred to as environments instead of servers. While it is possible for multiple environments to be hosted by the same physical machine, it is usually preferable to avoid this practice, in particular with regards to the production environment, which should be by itself and not shared with any of the other environments.

3.6. Technology

This section covers the technology employed by the project or generated as a result. The implementation of this simulation system involved working on three main types of components:

i) system infrastructure
ii) server-side software
iii) client-side software.

The system infrastructure consists of four servers, all based on Linux 2.6.32:
i) development environment that controls the development tools such as project management, version control, bug tracking, automatic build cycle, and documentation
ii) staging environment, which keeps the latest version of the simulation for testing purposes
iii) production environment that hosts the latest stable version of the simulator, and
iv) map server that provides extensive mapping capabilities for Australia.

These servers are VMWare virtual machines running on commodity hardware. This type of virtualised environment offers great especially demand-driven allocation of resources, at the expense of a small performance loss. The support software used in these servers is mostly open source. The web server used by the four servers is the well-established Apache Project management is provided by Trac while Subversion makes possible source version control. Project documentation is wiki-based through MediaWiki. The map server offers a WMF service through Geoserver and Geowebcache. Most of the layers used for the map of Australia were obtained from the Open-StreetMap thanks to their Creative Commons Licence and are stored by the geographical database

PostGIS.

The server-side software corresponds to the implementation of the simulation model described above. This software was written from scratch, its source code resulting in about 16,000 lines. The choice of language was Java 1.6 due to the adequate features of the language but more importantly the extensive number of support tools and associated technologies available both commercially and freely. The simulation model is implemented as a Web service that receives its configuration settings and simulation parameters as an input and returns a dataset of results. This web service runs within the context of the Web container Caucho Resin, which in turn sits on top of Apache. Handling of web request parameters is supported by the Model View Controller Web application framework Stripes.

Client-side software consists of about 4,000 lines of JavaScript code that run on the user browser. This software basically collects configuration settings and simulation parameters through its web-based user interface before using the server-side web service to run the simulation. After the simulation process has finished, the results dataset is parsed and formatted by the client-side software in order to generate detailed graphic and textual reports. A crucial part of this user interface is map navigation, which is achieved thanks to OpenLayers.

3.7. Evaluation

In this section, we provide a system (i.e. non-functional) evaluation of the simulator. This type of assessment is important for determining not only the quality of the simulation system in general (Williams and Smith, 1998) but its suitability to the target audience and the level of fulfillment of the requirements proposed. We chose a scenario-based approach (Kazman et al., 1996) to evaluate the simulation system, where several simulation scenarios, offering different degrees of complexity, are proposed and the corresponding system simulation results analysed.

These scenarios were the basis for three main types of experiments in the evaluation. Specifically, below we provide the evaluation of the simulator from the point of view of execution time of simulations using several ranges of simulated time and spatial resolution. The next subsection analyses the possible benefits of employing a sparse grid representation by the spatial model. The third type of experiments provided by the next subsection discusses the relationships between memory requirements in the server and both the length of the simulated time and the spatial resolution.

Simulation Scenarios

We selected two different simulation scenarios with different degrees of complexity in their definition:

Scenario 1: the simulation area comprises an area of about 9085 km2 (114.38 km by 79.71 km) containing a single observation of 100 adult fruit flies at a location in the centre of the simulation area, a 56 km2 orchard with 1000 trees of 200 fruits each about 11 km east of the observation, and another 236 km2 orchard with 2000 trees of 100 fruits each about 15 km west. The fruits have only a season, suitable for fruit flies, and the weather remains constant at 25.

Scenario 2: similar to Scenario 1 but in addition it contains a 164 km2 lake about 3 km south of the observation, a large 273 km2 orchard with 50000 trees of 500 fruits each located about 40 km northeast and containing 100 eggs in its centre, two more similar orchards located 35 km east and 37 km southeast respectively, a third group of 50 adult flies between these two orchards, and a 1 km wide river crossing from north to south about 15 km from the centre of the simulation area.

Execution Time

Execution time is a crucial non-function feature of a simulation system. In our case, execution time is directly related to the non-functional requirement of response time. We performed two different types of experiments to measure execution time in relation to the spatio-temporal nature of the simulator: i) measuring execution time based on how long the simulated period was, and ii) measuring execution time based on the spatial resolution. Before each of these experiments were executed, the Simulation Engine was reset in order to provide a fresh start for all experiments. Of course, no other simulations were taking place at the same time we performed these experiments.

Results of running both scenarios are reported in (Garcia Adeva and Reynolds 2011b).

The conclusions that can be drawn from these graphs are in some cases obvious. For example, that a more complex simulation scenario will take longer to run than a simpler scenario. There is no correlation between the number of simulated days done and execution time due to the stochastic nature of the simulator, while the differences are small. What may not be obvious at first is why execution seems to become faster with time (i.e. the curve is convex instead of concave). We believe there two element that contribute to this effect: i) the spatial model uses a sparse representation for grid cells which is expected to benefit longer simulated periods while shorter periods could be penalised due to grid cell instantiation overhead; ii) the simulation engine works on the Java Virtual Machine (JVM), which features a technology called HotSpot that provides adaptive optimisation by dynamically recompiling portions of a program, thus especially benefiting an algorithmic-oriented and repetitive program like in this case.

The results indicates that the larger the resolutions the longer it will take for the simulation to complete, with Scenario 2 always taking longer than Scenario 1 at same spatial resolutions.

Sparse Grid

Subsections above described how the spatial aspect of the simulator is represented by a discrete grid, and offered details on the implementation of the model in question by using a sparse representation of cells in the grid. In order to evaluate whether this approach is useful, we performed several experiments. We included how many cells had to be created over 100 days when simulating both Scenario 1 and Scenario 2 at four different spatial resolutions (50 by 35 cells, 150 by 105 cells, 300 by 210 cells, and 450 by 314 cells) and two different scenario sizes (normal size of 114.38 km by 79.71 km and an area four times this size).

The conclusions that could be drawn from these results include the apparent benefit of this sparse grid for large simulation areas, whereas the benefit is modest for smaller areas.

This type of outcome can be expected when we realise that the larger the simulation area, the more sparse its own features are. In this situation, a sparse representation of these features fits better their natural distribution. In contrast, the benefit is not so significant for smaller simulation areas. By observing the graphs of results, it seems that only short simulations of up to 50 days area benefited, whereas longer simulations tend to use all the cells in the grid anyway.

Memory

This section offers a set of simulation results with respect to memory usage in the server hosting the simulator engine. We were interested about whether either the length of the simulated time period or the spatial resolution had any significant effect on server resources that could eventually affect the reliability of the service. This consideration is important as per the requirement of availability.

(Garcia Adeva and Reynolds 2011b) provides a graphical view of how much memory (MB) remained available in the server after performing the simulation of both Scenario 1 and Scenario 2 over a period of 100 days for different spatial resolutions (50 by 35 cells, 150 by 105 cells, 300 by 210 cells, and 450 by 314 cells).

The main conclusion that we reached based on these experimental results is that it is difficult to control available memory due to the nature of the JVM where the simulation engine runs on. For example, there does not seem to be a direct relationship between spatial resolution and memory usage. Similarly, running the simulator for longer periods does not involve a trend towards growing memory requirements. This represents a fairly positive indication in order to achieve the requirement of scalability. However, there is a clear indication that the more complex and long the simulation is, the more active the JVM finds itself dealing with garbage collection (Dijkstra et al., 1978). In particular, the spatial resolution seems to produce the most impact in garbage collection activity. The obvious reason for this apparent relationship between garbage collection activity and length of simulation or spatial resolution is the huge number of memory objects being constantly created, especially by the spatial and dispersal models of the simulation engine.

3.8. Verification and Validation

With respect to validation in ecological simulation, the ideal method would consist of comparing results with the gold standard of high quality historical records. However, it is very rare to find such a comparison in the natural sciences (Sojda, 2007) and we will also not manage that high standard. This is probably due to this area of research often revolving around vague problems and suffering from uncertain and incomplete knowledge. We were able to locate a dataset from an outbreak of Bactrocera fruit fly that occurred in 1989 in Perth (Australia). The next subsection describes this event while the subsequent subsection provides details on an empirical comparison between the limited data that exists from the outbreak and the results of a simulated scenario using our simulator.

Outbreak Scenario

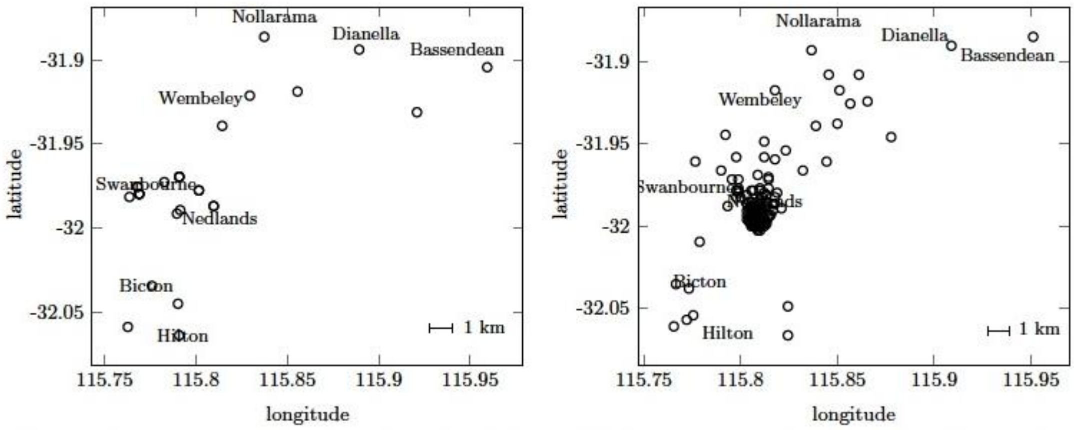
The first incursion of Queensland fruit flies (Qfly) (i.e. *B. tryoni*), in Western Australia was in February 1989. It spread rapidly throughout Perth (Bateman, 1989). The incursion was identified in tomato grown in the location of Dalkeith and collected on 16 February 1989, to emerge on 8 March 1989. Fruit fly numbers were highest in an area of 15 km2 surrounding the locations of Nedlands and Claremont. The initial infestation of *B. tryoni* was concentrated in an area of 15 km2 around the area Nedlands/Dalkeith, but after four months of trapping it covering an area 100 km2, where more than 200 *B. tryoni* flies had been trapped in the area (Sproul and Froudist, 1992; Yeates et al., 1992).

An eradication program commenced in August 1989, by which time trapping revealed the extent of the infestation to be over 125 km2 of the Perth metropolitan area, increasing to 270 km2 by 10 December 1989 and to 300 km2 by 23 February 1990. The eradication program used three stages of attack against this invasive pest. It incorporated a lure trapping system, insecticide baiting and release of sterile male fruit flies. Sterile fruit fly release started in January 1990 and continued until December of 1990 that year, after a successful eradication.

A system of fruit surveillance started in September 1989 and continued to June 1990. The procedure put in place had the intention of detecting new invading propagules in the field before the adults mature (Sproul et al., 1992).

By the time of commencement of the eradication program, trapping and fruit monitoring showed that wild Qfly numbers and fruit infested was declining significantly. Very few were found inside the original infested area in December 1989 and January 1990. However, even after eradication commenced, infestations were found outside the original zone mainly to the north and east, but also to the south of Dalkeith. Qfly were found by August 1989 infesting Innaloo and North Perth and in September spreading north-east along the Swan River and north of the original infestation; there was also one found in the south at Spearwood. By the end of October 1989 the infested area had expanded to include Doubleview, Innaloo, and the Mt Hawthorn/North Perth and Bassendean/Bayswater regions. Infested areas in December 1989 included Gosnells, Beckenham, Lesmurdie, South Guildford, Coolbellup/Hilton, Nollamara, Dianella, Carine, Manning, Victoria Park, East Victoria Park, Carlisle and Lathlain and fruit flies were found in the outer-Perth suburbs of Greenmount, Midland and Karrinyup by January 1990. On the 26th of January a single male fly was collected in the town of Northam, approximately 100 km north east of Perth. Northam was declared infested and the eradication program was extended to this area. The main concentration of wild Qfly captures during February 1990 was between the Swan River and the Canning River east to the airport. Fruit flies were also found outside the known infested areas at Darlington and Armadale (Sproul et al., 1992; Yeates et al., 1992).

The actual fruit fly detection in 1989 (Figure 4(a)) conforms to a latitude range from 31.9 to 32.1S and a longitude range from 115.7 to 115.9 E. Outermost examples included Wanneroo (27 km from Dalkeith), Karrinyup (12.4 km), Warwick (15.9 km), Mount Hawthorn (7.9 km), Bassendean (16 km), Midland (23.8 km), White Gum Valley (7 km), Beckenham (16 km), Victoria Park (10.5 km), Lesmurdie (23.1 km) and Pickering Brook (34.8 km). However the flies dispersed in clusters and sometimes chose an aggregation site and mating is facilitated.



(a) Trap detections from September – November 1989 after reported incursion.

(b) Fruit fly locations after simulating the incursion.

-

Figure 4: Comparison of actual fruit fly detections in the 1989 Perth outbreak versus a simulated scenario.

On 10 December 1989 the Perth area infested with Qfly had increased to 270 km2 and by 23 February 1990 it expanded to 300 km2. By this time, trapping and fruit monitoring showed that wild Qfly numbers and fruit infested was declining significantly. Very few flies were found inside the original infested area between December 1989 and January 1990.

Sterile fruit flies release started in January 1990 and continued until December that year hence ensuring a successful eradication. According to Sproul and Froudist (1992), fruit surveillance started in September 1989 and continued to June 1990. The procedure put in place had the intention of detecting new invading propagules in the field before the adults mature. The eradication program commenced in August 1989 and used three stages of attack against this invasive pest. They used a lure trapping system, insecticide baiting and release of sterile male fruit flies. *B. tryoni* was eradicated from the Perth Metropolitan area in 1990 using pheromone traps and sterile fruit y release (Sproul and Froudist, 1992; Fisher and Sproul,1984; Meats et al., 2001).

Qfly was eradicated from the Perth Metropolitan area in 1990 using pheromone traps and sterile fruit y release (Sproul and Froudist, 1992; Sproul et al., 1992, 2001). As a result of this incident, Western Australia spent $8 million between 1989 and 1990. Since completion of the eradication program, a small number of Qfly have been trapped, leading to several separate incursions. Consequently, a small declared outbreak was announced in 1995 (Sproul et al., 2001).

Simulated Scenario

Figure 4 provides a visual comparison of actual events described in the previous subsection along with the results of a simulated scenario. Figure 4(a) contains the actual cases of fruit flies detections in traps around Perth during the period September - November 1989. The extent of the invasion of breeding populations of *B. tryoni* was revealed by trapping and fruit surveillance before the commencement of the eradication program. Figure 4(b) corresponds to the simulation results obtained after trying to replicate the scenario at the time.

The highest number of *B. tryoni* was in the area surrounding Dalkeith, covering an area of 15 km2. This was the original area of infestation, although later the expansion extended to 100, 125, 200 km2, and so forth. The expanded infestation area was determined by trapping until June 1990. The reports from the one year of monitoring program (Sproul et al., 1992; Yeates et al., 1992) show that the *B. tryoni* population had survived Perth's winter season.

In a fruit survey, where fruit yielded maggots of *B. tryoni* in surrounding Perth metropolitan areas, infestations from north/western, north-eastern, south/western and south/eastern areas. The actual fruit fly detection in 1989 (Figure 4(a)) compared with the simulated scenario (Figure 4(b)) shows a great similarity in the range of infested area, with latitude range from -31.9 to -32.1 and longitude range from 115.76 to 115.95. Outermost examples included Wanneroo (27 km from Dalkieth) Karrinyup (12.4 km), Warwick (15.9 km), Mount Hawthorn (7.9 km); Bassendean (16 km) Midland area (23.8 km), White Gum Valley (7 km), Beckenham (16 km), Victoria Park (10.5 km), Lesmurdie (23.1 km), and Pickering Brook (34.8 km).

In order to perform a realistic simulation, the following hypothetical situation was considered; we assumed that one fruit (e.g. loquat) was infested with fruit flies since around the incursion area one female and six males were caught in traps. This probably meant that a breeding population was in the area. Hence, the incursion was established at the first location where fruit flies were trapped (i.e. Crawley, west of the city centre and north of the river), with a population of zero eggs, one larva, zero pupae, and seven adults. We considered obstacles whose scores ranged from zero to one, where one is the value for the inhospitable ocean (100% mortality), river or mountain (50% mortality), or lake (30% mortality).

Host seasonality and orchard area were also included where quantity of fruit trees in the area and number of fruit per tree were quantified. This included the suitability of the fruit and ripeness and how many days there are available (normally is considered days (e.g. 14); host suitability (e.g. excellent), ripeness (e.g. ripe); and second options is day (e.g. seven); host suitability (e.g. good), ripeness (e.g. over-ripe). The weather period is specified by the temperature and number of days, days (e.g. 70), temperature (e.g. 30_C); second option; day (e.g. 70), temperature (e.g. 20_C). The simulation area was 52.65 km by 27.72 km, with a grid of 100 by 53 cells, which corresponded to 1 km2 per cell. We ran the simulator for 170 to 210 days, in order to provide enough time for population growth, establishment and final dispersal.

The simulation showed that the infested area surrounding Crawley increased and flies were spreading into north-western, north-eastern, south-western and south-eastern areas in a very similar way to that which happened in 1989. Also, even when the weather period, specified by the temperature and period of days, was changed from 70 days, 30_C temperature to the second option of 70 days, 20_C temperature, or with an extra cold period added e.g. 30 days and 18_C temperature, the dispersal gave a similar picture about how

B. tryoni would spread.

These results tend to confirm what one would expect during an outbreak. The population parameters used relate to average climatic conditions prevailing during the normal infestation period. It should not be used outside the normal period of Q-y occurrence. In its native range overwintering adult flies become active in August, and by late summer to early autumn the flies are present in high populations (Ayling, 1989). Further calibration and verification would be needed, and possibly different modelling techniques. Fletcher (1973) has for instance indicated that *B. tryoni* has a tendency to congregate in well-wooded locations along creeks and rivers. Marked flies were also more frequently caught in traps in gardens and suburban areas than in non-cultivated areas the same distance from the orchard. The authors accept that a more sophisticated approach may be followed in another year or ten years, but will settle for going somewhere towards addressing some aspects now.

In summary, we believe that the simulator offered a sensible answer to the question that had been posed prior to this incursion. We believe that it served its purpose in that the simulation should be able to provide spatial and time spread information in order to enable a rapid response to an incursion, specifically showing where it are would or could spread, how fast it would spread and the area of spread if the option is to do nothing.

3.9. References

- Ayling, G. (1989). The Queensland fruit fly eradication campaign. Western Australia Journal of Agriculture, (30):159-162.
- Banerjee, S. (2004). Revisiting spherical trigonometry with orthogonal projectors. The Mathematical Association of America's College Mathematics Journal, 357:375-381.
- Bateman, M. A. (1989). Eradication Strategy of an invading population of Queensland fruit in suburban Perth. Technical report.
- Booch, G., Rumbaugh, J., and Jacobson, I. (1999). The Unified Modelling Language User Guide. Addison-Wesley, Reading, MA, USA.
- Dijkstra, E. W., Lamport, L., Martin, A., Scholten, C., and Ste_ens, E. F. M. (1978). On the-fly garbage collection: an exercise in cooperation. Commun. ACM, 21:966-975.
- Fisher, K. T. and Sproul, A. N. (1984). The methodology for the sterile insect technique against the Mediterranean fruit y in Western Australia. In Proceedings of the 4th Australia Appl. Ent. Res. Conf., pages 12-20.
- Fletcher, B. S. (1973). The ecology of a natural population of the Queensland fruit fly, Dacus tryoni. iv. the immigration and emigration of adults. Australian Journal of Zoology, 21:541-565.
- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Garcia Adeva, J., Botha, J., and Reynolds, M. (2011a). A Simulation Modelling Approach to Forecast Establishment and Spread of Bactrocera Fruit Flies. Ecological Modelling. In Press.
- J. J. Garcia Adeva and M. Reynolds. (2011 b) "Web-based Simulation of Fruit Fly to Support Biosecurity Decision-Making", submitted.
- Garland, J. and Anthony, R. (2003). Large-Scale Software Architecture. Wiley.
- Goodwin, B. J., Jones, C. G., Schauber, E. M., and Ostfeld, R. S. (2005). Limited dispersal and heterogeneous predation risk synergistically enhance persistence of rare prey. Ecology, 12:3139-3148.
- Haines, E. (1994). Graphics Gems IV, chapter Point in Polygon Strategies. Morgan Kaufmann.
- Hurrion, R. (1976). The design, use and required facilities of an interactive visual

computer simulation language to explore production planning problems. PhD thesis, University of London.
- Kachitvichyanukul, V. and Schmeiser, B. W. (1988). Binomial random variate generation. Commun. ACM, 31:216-222.
- Kazman, R., Abowd, G., Bass, L., and Clements, P. (1996). Scenario-based Analysis of Software Architecture. IEEE Software, 13(6):47-55.
- Meats, A. W., Clift, A. D., and Robson, M. K. (2001). Technical assessment of fruit fly outbreaks and suspension zones for Medy and *B. tryoni* in Australia. Technical report.
- O'Keefe and Robert, M. (1987). What is visual interactive simulation? (and is there a methodology for doing it right?). In Proceedings of the 19th Conference on Winter Simulation, WSC '87, pages 461-464, New York, NY, USA. ACM.
- Sojda, R. (2007). Empirical evaluation of decision support systems: Needs, definitions, potential methods, and an example pertaining to waterfowl management. Environmental Modelling and Software, 22(2):269-277.
- Sproul, A. N. and Froudist, S. (1992). Fruit Surveillance for Qfly. Technical report.
- Sproul, A. N., Broughton, S., and Monzu, N. (1992). Queensland fruit fly eradication campaign. Technical report.
- Sproul, A. N., Broughton, S., De Lima, F., Hardie, D., Monzu, N., and Woods, B. (2001). The fight against fruit flies in Western Australia. Technical Report 4504.
- Williams, L. G. and Smith, C. U. (1998). Performance evaluation of software architectures. In Proceedings of the 1st International Workshop on Software and Performance, pages 164-177. ACM.
- Xie, H. and D., Y. P. (2006). Developing a Web-based System for Large-Scale Environmental Hydraulics Problems with Application to Oil Spill Modeling. Journal of computing in civil engineering, 20(3):197-209.
- Yeates, D. K., Yeoh, P. B., and Rechichi, S. N. (1992). Trap monitoring for *B. tryoni* in Queensland fruit fly eradication campaign. Technical report.

4. Implications for stakeholders

The main implication for stakeholders is that this project has successfully demonstrated a ‘proof of concept’ of a fast, flexible, usable simulator to allow real-time prediction of the likely outcomes of real or hypothetical EPP outbreaks. This allows biosecurity emergency response managers to quickly get an idea of the likely speed and extent of an outbreak and to ‘play’ with alternative biosecurity responses.

For developers of alternative, or follow-on EPP simulation software, the project demonstrates real-time simulation via a convenient web-based architecture. Calculations are done for any user with any computing device using the latest version of the simulation software on quick and powerful servers with access to local land-use data.

The underlying models use the current best insect and disease models but are flexible in allowing parameters to be set by local experts. The software has been ‘personalised’ to several example EPPs and a match of prediction with historical outbreak data has been achieved.

The main implication is that this pathway of robust, easily extensible general simulators, accessed via a web-server architecture is a very viable option for future development.

5. Recommendations

Recommendations based on this research fall into three categories: 1) potential users to take up opportunities opened up by the developments here; 2) suggestions for further development of the tools and techniques here; and 3) more general lessons for biosecurity.

Under the first heading we have seen that the techniques in simulator development undertaken here have provided a proof of concept for a flexible, usable EPP spread prediction tool framework. Efforts should be made to explore ways of improving awareness of the availability of such tools amongst potential users. However, it must be emphasized that the products of this project are not quite ready to be used in the field as they are now.

Thus, under the second heading, we can collect suggestions for polishing the tools for field use in the near future as well as a large number of specific improvements and extensions that we have already noted. Field trials of the tools during actual outbreaks are required as is the collection of more general feedback on new functionality and usability issues. Decisions about how to run the server and maintain the software and/or about commercialisation need to be made.

Further development of the simulation technology include 'personalising' it to more pests.

Other possibly desirable developments are to adapt it properly for running on mobile phones and tablets, to better support a 'war game' type of simulation where parameters can be changed at intermediate stages, more robust import of needed land-use and weather data from known internet sources, and extensions to cover Asian neighbour countries.

Under the final category we include a recommendation for further research into data for validation or verification of simulators. Details of trap data over time, as well as detailed accounts of land use, weather patterns, human activity during outbreaks need to be collected and made available in a publicly usable way. Our validation activities were certainly hampered by the sketchiness of such data.

6. Abbreviations/glossary

ABBREVIATION	FULL TITLE
CRCNPB	Cooperative Research Centre for National Plant Biosecurity
EPP	Emergency plant pest
UWA	The University of Western Australia
DAFWA	WA State Government Department of Agriculture and Fisheries
CSSE	School of Computer Science and Software Engineering (at UWA)
Qfly	Queensland fruit fly
JVM	Java Virtual Machine (runs Java programs)

7. Plain English website summary

CRC project no:	CRC10073

Project title:	Surveillance Simulation Platform
Project leader:	Professor Mark Reynolds
Project team:	UWA CSSE: Mark Reynolds, Juan Jose Garcia Adeva, DAFWA: Darryl Hardie, John Botha, Maria Majer
Research outcomes:	▪ A surveillance prediction simulation platform for validating surveillance strategies; ▪ novel landscape-level modelling techniques for pest spread simulation; and ▪ validated simulation technology using historical emergency plant pest incursion data.
Research implications:	This pathway of robust, easily extensible general EPP simulators, accessed via a web-server architecture is a very viable option for future development, commercialisation and/or roll-out.
Research publications:	**Press** ▪ Simulator will aid in pest defence. Good Fruit & Vegetables, November, 2010. ▪ Pest scenarios test responses ▪ Simulator to aid protection from exotic plant pests ▪ Spreading the collaboration ▪ From storage to export: learning about the grain supply chain **Conferences** ▪ 'Awareness of the Asian Gypsy Moth: a Threat to Australia's Eucalyptus Plantations and a Concern for Restoration Programs', SERI World conference on Ecological Restoration, Perth, 2009. ▪ 'Modelling establishment and spread potential of Bactrocera fruit flies: Australian concerns for a surveillance program', Australian Entomological Society, Darwin, 2009. ▪ 'Development of a Surveillance Simulation Platform for establishment and spread of EPPs: examining Australian concerns', Australian Entomological

	Society, Darwin, 2009. ▪ 'Modelling the establishment and spread of emergency plant pests (EPPs) in Australia: simulate or suffer', Global Biosecurity, Brisbane, 2010. ▪ 'Modelling the spread of fire blight', Global Biosecurity, Brisbane, 2010. ▪ 'Simulation technology to predict the establishment and spread of Asian gypsy moth, Lymantria dispar asiatica', Australian Entomological Society, Perth, 2010. ▪ 'Simulation of Honeybee Nectar Foraging for Determining Effects on Local Flora', Modelling and Simulation Society of Australia and New Zealand, Perth, 2011. **Journals** ▪ 'Modelling establishment and spread potential of Trogoderma granarium Everts: Australian concerns for a surveillance program', Resistant Pest Management Newsletter, Vol. 18, No. 2, pp. 4 - 6, Spring 2009. ▪ 'A Simulation Modelling Approach to Forecast Establishment and Spread of Bactrocera Fruit Flies', Ecological Modelling, accepted 2011 in press. **RECENT SUBMISSIONS under REVIEW** • Threat of Asian gypsy moth, Lymantria dispar (L.) (asiatica and japonica) to Australia: Motivation for developing simulation technology. (Being updated for resubmission, Dec 2011) • Web-based Simulation of Fruit Fly to Support Biosecurity Decision-Making. (Submitted to Ecological Informatics June 2011) • Simulation Modelling of Nectar and Pollen Foraging by Honeybees. (Submitted to Biosystems Engineering September 2011) • Web-based Simulation of Nectar and Pollen Foraging by Honeybees. (To be submitted to Decision Support Systems, Dec 2011)
Acknowledgements:	The research team acknowledges the CRC for National Plant Biosecurity (CRCNPB) for provisions of funding for this project.

	The research team also expresses sincere thanks to many other researchers at UWA, DAFWA and CSIRO who have assisted with knowledge of the behavior of many pest species and with advice and feedback about the usability of the system.