3Worlds, a simulation platform for ecosystem modelling

Jacques Gignoux, Ian D Davies, Shayne R Flint

3Worlds, a simulation platform for ecosystem modelling

Jacques Gignoux[1], Ian D. Davies[2], Shayne R. Flint[3]

1 Institute of Ecology and Environmental Sciences of Paris (iEES-Paris),
Centre National de la Recherche Scientifique,
Sorbonne Université,
4 place Jussieu,
75005 Paris, France

jacques.gignoux@umpc.fr

2 Research Institute for the Environment and Livelihoods,
College of Engineering, IT and Environment,
Charles Darwin University,
Darwin, Northern Territory 0909 Australia
and
Fenner School of Environment and Society,
The Australian National University,
Canberra ACT 0200 Australia

3 School of Computing,
College of Engineering and Computer Science,
The Australian National University,
Canberra ACT 0200 Australia

submitted to *Ecological Modelling* as an **original research** paper

(**8738** words, including tables, figure legends and bibliography)

Running head (**34** char): *3Worlds: simulating any ecosystem*

Version 15/06/2022

Highlights

- We present *3Worlds*, a new modelling platform that represents any ecosystem as a dynamic graph.
- Ecological meaning of the graph is defined by adding descriptors and selecting from 10 possible atomic transformations to its nodes and edges.
- Model configurations are also graphs which largely reduces model comparison to a comparison of graphs.
- The software enables modellers to find an appropriate level of abstraction for their problem by accelerating model construction and facilitating model comparisons.
- By focusing on system representation rather than simulation technique, modellers can implement discrete event, multi-agent, system dynamics and individual-based models within the one platform.

Abstract (**280** words)

Ecology, like many disciplines, commonly relies on simulation to provide insights into the dynamics of complex systems. Yet there are two unresolved problems for ecological studies relying on simulation. First, it is often the case that simulators representing the same system, designed for ostensibly the same purpose, differ in their results with the reasons buried deep within computer code. Second, ecology is a diverse discipline and each sub-discipline necessarily has its particular simulation methods. This raises a problem as to how models from these various fields can be coupled for transdisciplinary studies. We built a new simulation platform named 3Worlds, grounded on a concept familiar and common to all fields of ecology: the ecosystem. We defined the ecosystem for the purpose of simulation by a precise set of rules. The platform can implement models from fields as diverse as food web, population and landscape ecology, energy and material stocks and fluxes, and techniques such as agent-based, cellular automata and discrete-event simulation. In addition, we developed a dynamic graph to represent ecosystems as a set of interacting components. Our approach goes some way to unifying ecology for the purpose of simulation and reduces the problem of code comparison to a comparison of two graphs: (1) a specification graph that complies with the rules of what constitutes an ecosystem, and (2) the successive graph states of a particular simulation trajectory representing the ecosystem. Two applications constitute the core of 3Worlds. *ModelMaker* builds the ecosystem compliant model and *ModelRunner* executes the model represented as a dynamic graph. A library of ~24 models illustrates how 3Worlds can simulate very different systems, from simple 1-equation 1-variable models to individual-based systems with thousands of ecosystem components.

Keywords

ecosystem, dynamic graph, emergence, simulation, model comparison, multiple scales

1. Introduction

Ecology and other sciences use simulation models (or *simulators*) to study systems where experimental manipulation of those systems is either impractical, unethical, dangerous or intractable. Despite the adage '*To predict is not to explain*' (Thom & Noël, 1993), as exploratory tools, simulation models are used to do both: they are used to predict the future of ecosystems under changing circumstances, and to explain real-world observations.

Problems arise however, when simulators representing the same processes yield different outcomes for the same case study, as has been shown by many model intercomparison studies (Bugmann et al., 1996; Cary et al., 2006; Friedlingstein et al., 2006; Gritti et al., 2013; Hantson et al., 2020; Jepsen et al., 2005; Melilo et al., 1995; Roxburgh et al., 2004; P. Smith et al., 1997). Although formal methods exist to assess the significance of differences between model outputs (J. Smith et al., 1996) and to verify the individual models (Woodcock et al., 2009), tracking the cause of the differences between independently developed models or even versions of the same model remains an open problem. This contributes to the lack of confidence in findings derived from simulation modelling (Lenhard & Winsberg, 2010), despite their crucial role and the time invested in their development.

The problem is particularly apparent when the simulators are derived from the same dynamic equations (e.g. Lim & Roderick, 2009). The difficulty of identifying the causes of differences in model outputs results in variation in output being attributed to random error and subsumed within estimates of uncertainty (Intergovernmental Panel on Climate Change, 2014). This would seem a missed opportunity to consider these differences as informative.

This need not be so. All simulators in ecology deal with some kind of ecosystem representation, and thus should share a common conceptual background. But most often, their design is method-based rather than concept-based, leading to incompatible modelling worlds: differential equation systems (Gurney & Nisbet, 1998), cellular automata (Ermentrout & Edelstein-Keshet, 1993; Favier & Dubois, 2004; Hernandez Encinas et al., 2007; Muci et al., 2012), individual-based models (Grimm & Railsback, 2005) and multi-agent systems (Amouroux et al., 2009; Bellifemine et al., 2001; Minar et al., 1996; North et al., 2007; Wilensky, 1999). We argued (Gignoux et al., 2011) that the ecosystem concept, as proposed by Tansley (1935), was well adapted to simulation modelling: the ecosystem is a *multi-aspect*, *scale-independent*, *observer-selected* and *recursive* object. We further proposed (Gignoux et al., 2017) that a dynamic graph (Harary & Gupta, 1997) is a representation applicable to any *hierarchical system*, i.e. any system able to display the emergent properties characteristic of complex systems. We captured these ideas in a set of rules, hereafter called a *specification archetype* (Flint, 2006), describing any ecosystem. The interest of such a method for model comparison is obvious: since the path from abstract knowledge to code is explicit and recorded, differences in implementation can be traced and analysed.

3Worlds is new software for researchers interested in modelling any aspect of ecosystems. It is named after the lithograph 'Three Worlds' by M.C. Escher.

2. Software design

Most simulation models in ecology are developed for a single application, although they often require considerable programming skill (e.g. individual-based or agent-based models: Bousquet & Le Page, 2004; Dorri et al., 2018; Ferber, 1995), a skill in which ecologists are not necessarily trained. Scientists spend ~30 % of their time writing code rather than doing the science in which they are expert (Hannay et al., 2009). Many platforms already exist to address these issues by providing a formal framework to guide and accelerate development time (GAMA: Amouroux et al., 2009; ASCEND: Piela et al., 1991; VLE: Quesnel et al., 2009; STELLA: Richmond et al., 1987; NETLOGO: Wilensky, 1999; DEVS: Zeigler et al., 1997). However, to the best of our knowledge, all these platforms focus on the simulation technique they implement rather than the domain of the system they represent: system dynamics (Richmond et al., 1987), discrete event simulation (Zeigler et al., 1997) and the multi-agent paradigm (Bonasso et al., 1997) are examples.

2.1. The archetype: rules for modelling ecosystems

Rather than imposing a specific simulation technique, 3Worlds focuses on the concept of the ecosystem as it applies to ecological simulation. In practice, this means we have developed an *ecosystem archetype* (Appendix 1): a list of rules specifying, among other things, the structure and dynamics of the ecological model. For example, some rules describing a properly formed specification for dynamics of a simulator are: (i) a simulator must have at least one process; (ii) that process must have at least one function chosen from one of ten function types; (iii) all processes must have a common conception of the passage of time; (iv) processes can define the order of execution if they occur simultaneously… and so on. While the archetype is a large document, we have been careful to maintain generality and avoid over-specification, allowing, to the best of our knowledge, any model to be proposed. Thus, while models can be arbitrary, their design is not because their specification will necessarily comply with the specification archetype. In a general sense, the specification archetype is a drawing together of all knowledge required from potentially diverse fields for the specific purpose of building an ecological simulator and follows the methodology of *aspect-oriented thinking* (Flint, 2006) (Appendix 1).

Software architectural and code generation concepts are also used to manage code for ecological processes (e.g. growth, reproduction, survival, environment dynamics functions) by automatically generating code that modellers modify with their preferred formulations (a solution formerly tested in the MUSE simulator: Gignoux et al., 1998). This limits program coding to just those parts relevant to the modeller.

3Worlds includes a tool called *ModelMaker* (Appendix 2) which can be used to form specifications for ecological simulations in accordance with the 3Worlds specification archetype (Fig. 1). Another tool, called *ModelRunner* (Appendix 2), can then be used to run, analyse and document these simulations. *ModelRunner* can also extract analytical data that can be used to compare models in terms of clearly defined ecological concepts. Separating the two phases of simulator construction and execution (Appendix 3) adds confidence that we are running, analysing and documenting simulations underpinned by the same shared concept of what constitutes an ecosystem.

2.2. Representing the ecosytem as a dynamic graph

3Worlds uses a *dynamic graph* (Harary & Gupta, 1997) to represent ecosystems at any spatial, temporal and organisational scale. The ecosystem is a graph (Gross & Yellen, 1999), its components, whatever they are in reality, are *nodes*, and their relations are *edges*. Because the graph is dynamic, nodes and edges can be created or deleted during a simulation (Fig. 2). All graph elements (nodes, edges, but also the graph itself) carry *descriptors*, used to characterise their state at any instant in time (Gignoux et al., 2017). Relations (edges) can be of any kind, including a hierarchical relation describing the complex nesting of sub-systems. This provides an elegant solution to the apparent complexity of ecosystems: it allows for various types of *emergence* (cf. discussion in Gignoux et al., 2017), enables the comparison of system structures and simulation trajectories, and can represent virtually anything an ecological modeller can propose.

Representing the modelled ecosystem as a dynamic graph has the benefit of reducing the possible ways a system can change to just ten atomic graph operations (Fig 2).

2.3. Dynamics

What makes the ecosystem graph dynamic are ecological *processes*. Processes apply to families of nodes sharing common descriptors (e.g., members of an animal species, landscape units of a certain type, etc.). They enable modification of descriptors of a single node or a pair of nodes linked by an edge, and the deletion or creation of nodes or edges. Process interactions are scheduled by *timers* that manage regular (recurring), irregular (event-driven) or predetermined (scenario) time steps. Timers can handle time units from microseconds to millennia, spanning a wide range of scales covering most problems studied by the numerous fields of ecology. Many timers can interact during a 3Worlds simulation, based on the most relevant time representation for each simulated process.

In most modern multi-agent systems, the strong autonomy of agents translates to assuming no simultaneous events exist, and this is modelled through a random order of activation of agents at each time step (e.g. GAMA: Amouroux et al., 2009). This is incompatible with computing exact resource budgets (usually matter and energy) where agents share a common resource. As with (parallel) DEVS (Chow et al., 1994), the 3Worlds simulator manages the simultaneity of events. Due to the analytical decomposition of natural phenomena into separate processes, simultaneous processes may actually be linked by causal relations, in which case the cause must be computed before its consequence, even if they occur within the same time step. For example, resource uptake must precede growth. It is important that modellers are able to decide in which order to compute simultaneous processes to satisfy the logic of their causal analysis of processes. Interaction between processes and graph components is handled by our re-implementation of the *rendezvous* system from the Ada programming language (**rvgrid** library).

2.4. Structure

In an individual-based model (Grimm & Railsback, 2005), every individual differs from all others. But it is common practice to assume that some groups of individuals share some things in common. This is the essence of modelling: finding commonalities within an ocean of particular cases. We use the concept of *category* to group system components that ‘look like each other’ in some way. In the 3Worlds dynamic graph, categories are used to specify (examples based on two categories, *plant* and *animal*):

1. common descriptors to groups of components (e.g. a *plant* species average growth rate, an *animal* cohort survival rate, etc.);
2. functions that operate on a group of components of that same category (e.g., *plant* growth differs from *animal* growth. Fig. 3);
3. which *type of relation* is possible between components of different categories (i.e. *herbivory* is an *animal* → *plant* relation).

This category concept is similar in many ways to the class concept used in the UML and in object oriented programming: categories group data to indicate what 'type' or 'class' a component is. Categories can be nested, and partitioned into sets of mutually exclusive categories (e.g. to define something as either a *plant* or *animal* but not both: Fig. 4). They are central to the organisation and execution of a simulation in 3Worlds as they formalise the coexistence of entities with completely different characteristics and behaviours, something familiar to ecologists manipulating all sorts of classifications such as taxons, trophic levels, development stages, functional groups and so on.

2.5. Space

Ecosystem models may be spatially explicit, i.e. space may be required to compute interactions between ecosystem components. In 3Worlds, we provide some state-of-the-art spatial algorithms like optimised space searching using *k*-dimensional (k-d) trees (Samet, 1984) and a variety of methods to manage edge-effects in spatial models (Table 1). We re-implemented and generalised the k-d tree algorithm proposed by P. Toivanen (https://dev.solita.fi/2015/08/06/quad-tree.html) in the **uit** library.

2.6. Abstraction

3Worlds can produce models at any temporal, spatial and organisational scale. This enables researchers to test the effect of the detail of system representation on simulation outputs. This is rarely done (e.g. Davies, 2014, for an extensive study of spatial and temporal scale effects in fire propagation models) although it has long been known that not only scale, but also the level of detail in model construction, has significant effects on simulation outputs (e.g. Gauzens et al., 2013). This has been formalised in abstraction theory (Zucker, 2003): the more abstract a model, the less detail it has. 3Worlds makes it possible to quickly check the effect of abstraction on model outputs, thus enabling selection of an appropriate level of abstraction for the question at hand.

2.7. Outputs

All simulation platforms provide some means of visualising or saving the state of the simulation at any time. In 3Worlds, any descriptor at any hierarchical level of any of the components of the dynamic graph, whether they persist throughout the simulation or are ephemeral, can be tracked and sent to a versatile, customisable, graphical interface providing quick runtime feedback. A library of visualisation objects (graphs, maps, time series) building on the scientific charting chartfx library (Steinhagen et al., 2019) can be freely assembled to adapt outputs to the needs of the researcher. Simulators can also be run with file output alone for unattended deployment on local or remote systems.

2.8. Experiment design

Once a simulator is ready to run, it will be subjected to experiments of various designs (Kleijnen et al., 2005; Peck, 2004) to provide insight and publishable results. We currently handle simple factorial and sensitivity analysis experiments, but plan to integrate 3Worlds with OpenMole (Reuillon et al., 2013), a platform specially designed for managing big simulation experiments (including deployment on supercomputers, clusters, grids, etc.) in the near future.

3. Comparing models

Our overall aim in developing 3Worlds is to enhance confidence in the knowledge gained from simulation. Confidence can be traced through the following chain of reasoning:

- The *3Worlds Specification Archetype* (3WSA) is, in effect, a (meta-) specification for 3Worlds model specifications. This archetype has been **validated** against Tansley (1935) in the sense that it is presented as peer-reviewed public statements (Gignoux et al., 2011) – a paper that can be discussed and challenged by experts as required.
- 3Worlds model specifications are formed using *ModelMaker*. Because *ModelMaker* enforces the 3WSA, we have some confidence that all 3Worlds models are well formed using clear concepts – that is, they are (automatically) **verified** against the archetype.
- All software is **validated** in some sense, by stakeholders agreeing that the specifications correctly reflect their modelling requirements. There is no automation here; it is often a cycle of testing and improvement, but the process is eased in 3Worlds because the specifications are **verified** against a clear set of ecological and simulation concepts captured in the archetype (in effect agreed among experts).
- Because *ModelRunner* executes every specification in the same way, and because every specification has been verified by *ModelMaker*, we can say that every running simulation is also **verified**. That is, every running simulation does what its specification says it should do (assuming *ModelRunner* has no bugs – if it does, they will impact all simulations and fixes will fix all simulations).

It follows we have verification all the way from the archetype to the running code, and therefore, a very good start for model comparison because we can (increasingly) trust *ModelRunner* to correctly implement every specification in the same way every time and we can (increasingly) trust *ModelMaker* to ensure that every specification is well formed in terms of well defined ecological concepts.

This means that *we can compare models by comparing their specifications*. As noted above, this is a human process, but because the specifications are properly formed using a clear set of concepts captured in the 3WSA, comparison of specifications is simplified and can be supported by the generation of statistics and documentation.

Model comparison is generally difficult because models are not usually developed with comparison in mind. This is because specifications for the systems to be compared may be represented in different forms: from nothing, through informal sets of ideas, to something written down in natural or formal languages. If the specifications are written in a formal language, they might be compared using automation and expertise. However, we would still need to verify the software against the

specifications using one or more of the established techniques and validate each specification against the needs of the modeller.

Comparing model structure is a way to better understand how models work, and hopefully the systems they represent. Model comparison is more easily done in 3Worlds by experiments testing various modifications to the (verified) specification graph to see their effect on the dynamics. Model comparison can also improve confidence in models by giving arguments for considering a model as a reference rather than, for example, considering the average of current models as a reference (Intergovernmental Panel on Climate Change, 2014).

Classifying models based on metrics can provide a valuable framework for comparing models (Keane et al., 2004). 3Worlds automatically computes statistics on the model's specification graph (Table 2). These allow comparison of models in a standard way: some of these quantities measure model size (e.g. the first column of Table 2), others measure its complexity (last column of Table 2). Other more elaborate data can be extracted, like the category tree or the main execution loop flow diagram, that would enable a much finer assessment of model differences. These data may seem trivial to collect, but they were once so difficult to find in the literature that a specific model description standard was developed to improve model publication practices (ODD: Grimm et al., 2006, 2010). *ModelRunner* can generate all the quantitative data needed to write the ODD description of a model: all the data of Table 2, but also pseudo-code and flow chart of the main iteration loop, user code, code structure (which ecological functions were implemented), entities modelled and spatial representation used.

4. Examples of applications

These examples have been developed to illustrate the versatility of 3Worlds. As such, even though some are based on field data, they do not constitute full ecological studies, which is not the focus of this paper. Full details of the models can be found in appendices 4-7.

When setting up a model in 3Worlds, the modeller has to (1) define all model entities, variables, constants, processes, scheduling rules, as well as outputs and inputs, by building a c*onfiguration graph* with the *ModelMaker* application; (2) write ecologically meaningful java '*user code'* in the process templates produced by *ModelMaker* at step (1). Appendix 3 details all the steps of a model building / simulation experiment session with 3Worlds in full.

4.1. A system dynamics model: testing the intermediate disturbance hypothesis

Purpose - The intermediate disturbance hypothesis (IDH: Connell, 1978) states that an appropriate disturbance regime can maintain a high local diversity when the disturbance is not 'too rare' or 'too frequent', but inbetween theses 'extremes'. The rationale is: (1) during primary succession, species richness is low at the beginning (only pioneer species), high during a dynamic species replacement phase, and low at the end when a few dominant species have eliminated most competitors; (2) we assume the existence of randomly occurring disturbances that kill most of the community, reverting succession to its initial stage; (3) if the disturbance frequency is very high, the community will stay in its early, low-diversity stage most of the time ; if it's very low, it will spend most of its time in its late, low-diversity stage ; when frequency is *intermediate*, the community will spend most of its time in its dynamic phase with maximal diversity. The IDH makes a very nice simulation

experiment to run for a student training session: (1) does it really work? (2) for what frequency? and (3) can we find a general link between disturbance frequency and community diversity?

***Entities, state variables and scales* -** To implement this, we use a competition Lotka-Volterra model with many species (40 at most in our example), coupled with disturbances which occur at irregular intervals by resetting population sizes close to zero. We used the Euler explict solving method supplied by 3Worlds. For more elaborate methods, it is possible to integrate the solvers of the org.apache.commons.math3 library into the user code. In 3Worlds terms, there are:

- 2 *component types*, the *community* and the *disturbance*;
- 3 *permanent* instances of them, one community and two disturbances;
- 1 *relation type*, with two instances, relating *disturbance* to *community*
- community variables (*drivers*) consist of a table of 40 population sizes *x*[40] and a single number, the time since last disturbance *tsd*;
- community has 3 tables of constants: the specific growth rates *r*[40], the carrying capacity for each species *K*[40], and the interspecific competition coefficients *alpha*[40,40].
- disturbance has two constants: frequency *freq* and intensity *inten*.
- the time scale and time step are arbitrary;
- the model is non spatial;
- simulation stops after 1000 time steps

As a result, the simulated dynamic graph is very simple: it only has 3 permanent nodes (1 community and 2 disturbances) and at most two ephemeral relations (from each disturbance to the community). Only the descriptors and relations are dynamic.

***Process overview and scheduling* –** There are three ecological processes in this model:

- community growth (*commGrowth*, of type *changeState*, cf. fig. 2) applies the Lotka-Volterra equations to the community;
- disturbance occurrence (*distOcc*, of type *relateToDecision*, cf. fig. 2), uses random numbers to decide when a disturbance event occurs;
- disturbance effect (*distEffectComm*, of type *changeOtherState*, cf fig.2) reduces the population sizes when a disturbance occurs.

Since there are two disturbance instances, the community is subject to two different regimes of perturbation. Scheduling and java code for these functions can be found in appendix 4.

***Details* –** Full configuration graph, variable list, flow chart, user code and a screen copy can be found in appendix 4. Model metrics are in Table 2.

***Conclusion* –** This model is among the simplest ones that can be implemented within 3Worlds.

4.2. An agent-based model: *Boids*

***Purpose* –** The Boids model was initially proposed by Reynolds (1987, 1999) to demonstrate that ‘complex’ behaviour could emerge from simple interactions between autonomous agents. It

simulates flocking behaviour of animals like birds or fish when flying or swimming together in a seemingly coordinated way. The test of the self-organisation of the flock/school is essentially visual: when starting from a random distribution of animals, after a while they should organise into clumps that move together. The precise algorithm we used for this toy model was found here: https://betterprogramming.pub/boids-simulating-birds-flock-behavior-in-python-9fff99375118.

Entities, state variables and scales – We used an individual-based representation of the population, with identical individuals located in a 2D-space. Each individual detects other animals within a sight range and maps its movement to its neighbours. In 3Worlds terms, there are:

- 1 *component type*, the *bird*, with 250 instances at simulation start. Since the simulation time is short relative to their lifespan, they are *permanent*;
- 1 ephemeral *relation type*, called *sight*, for which instances are established between any two birds whenever they come within a detection range from each other;
- 6 bird driver variables: *x* and *y* coordinates of position in space, velocity in x and y directions (*vx*, *vy*) and acceleration (*ax*, *ay*);
- 8 bird decorator variables: bird flock barycentre (*sumX*, *sumY*), average velocity (*sumdX*, *sumdY*), movement to avoid collision with neighbours (*avoidX*, *avoidY*), distance to neighbour (*sepX*, *sepY*);
- 5 constants shared among all birds, hence attached to the whole system (called the *arena* in 3Worlds): minimal distance to maintain between birds (*safetyRange*), maximal attraction force (*maxForce*), radius of the local group (*range*), visual field extent (*visualFieldAngle*), and maximal flight speed (*maxSpeed*);
- the time step is 1s and the time extent is unbounded;
- the space is represented as a flat rectangular 2D continuous surface of 1000 × 1000 m. Space definition includes a search radius parameter, here 100 m;
- simulations are run until user intervention.

The resulting simulated dynamic graph comprises 250 permanent bird components with ephemeral sight relations between birds that are within detection distance. As before, only descriptors and relations are dynamic. Since bird drivers comprise locations in space, they can be represented as moving items on a map.

Process overview and scheduling – There are four ecological processes in this model. Notice that a process runs a full loop on all relations or on all components:

- *detect* (of type *RelateToDecision*, fig.2) searches if a bird is within the *searchRadius* of another. If so, a *sight* relation is established between the two making it a member of its local flock;
- *follow* (of type *maintainRelationDecision*, fig.2) decides if a bird, already related through *sight* to another one, maintains that relation, i.e. remains in its local flock;
- *prepareMove* (of type *changeRelationState*, fig. 2) loops on all neighbours (= all birds related through a *sight* relation) of a bird to compute their barycentre, average speed, etc., and stores the values in the bird decorators;

- *move* (of type *changeState*, fig. 2) applies the boids rules (cohesion, separation and alignment) to the bird based on its current driver and decorator values;

Unlike most current multi-agent implementations, computations are made simultaneously for all birds, i.e. state change (new position) only takes place after the *move* loop on all birds. Scheduling and java code for these functions can be found in appendix 5.

Details – Full configuration graph, variable list, flow chart, user code and a screen copy can be found in appendix 5. Model metrics are in Table 2.

Conclusion – This model illustrates emergent properties (a complex coordinated movement) in a system of simple agents (birds). It shows 3Worlds can implement multi-agent systems, on the condition that they do not modify their state instantly (they must do it synchronously at the end of a time step, i.e. the system state is kept time-consistent).

4.3. A cellular automaton model: the '*Rabbit Rules*' fire spread model

Purpose – Fire spread is difficult to model realistically, specially when one wants to precisely predict area burnt, due to threshold effects in the combustion process that cause increasing large variations in prediction with time (Davies, 2014). Achtemeier (2003) proposed an original model as an intermediate between the pure physics model, so heavy that they are actually slower to run than the process they represent, and empirical cellular automata where temporal and spatial resolution affect the final results. The model makes an analogy between fire and 'rabbits' that 'eat' the fuel and jump to neighbouring fuel cells, this to represent spotting, a major source of imprecision in fire spread modelling. Agent-based in its design, this model can be implemented as a cellular automaton, and has been coupled with atmospheric physics to predict smoke plumes and feedback of fire on wind field (Achtemeier, 2013). The goal of this model was to empirically but correctly simulate the effect of spotting on fire spread, and it was heavily tested against field data. We re-implemented it in 3Worlds as a pure cellular automaton to illustrate this type of modelling.

Entities, state variables and scales – Everything takes place on a rectangular grid of cells which can take three states: unburnt, burning, burnt. In 3Worlds terms, there is:

- no component type – everything takes place in the *arena* component which represents the whole system;
- 2 main driver tables of dimension 300 × 300 cells of 10 m that represent the simulated system: *fuelBed* and *windMap*. These tables in turn contain fields that characterize every cell;
- fuel cells are described by 7 variables, among which *fuelType* and *fuelLoad* are the most important;
- 2 other drivers characterize the system, the *areaBurnt* and the number of burning cells (*nRabbits)*;
- wind cells contain 2 variables, the *x* and *y* components of wind *velocity*;
- 1 constant table of dimension 300 × 300 cells represents the topography through a *slope* factor with *x* and *y* components;

- other constants include spatial parameters (site dimensions, cell size) and fire parameters: average fuel height per fuel type, flame lifespan as a function of fuel type, wind effect on spotting distance, etc...
- the time step is 1s and the time extent is unbounded;
- the model does not use 3Worlds spatial features although it is spatial in nature;
- simulations are run until user intervention.

The simulated dynamic graph is reduced to the simplest: no nodes, no relations – only the system as a whole (an empty graph) exists.

Process overview and scheduling – Only one processes is required to run this model:

- *burn* (of type *changeState*, fig. 2) applies all the changes to the fuel grid

Scheduling and java code for this function can be found in appendix 6.

Details – Full configuration graph, variable list, flow chart, user code and a screen copy can be found in appendix 6. Model metrics are in Table 2.

Conclusion – Although this model makes very little use of 3Worlds capabillities, except concerning data structuration, its complexity is relatively high as the user code is quite elaborate compared to previous examples.

4.4. An individual-based vegetation model: the Lamto palm tree dynamics model

Purpose – This model illustrates the strong link a simulation model can have with field data by synthetizing various papers dealing with the population dynamics of *Borassus aethiopum*, a palm tree from West African savannas (Barot et al., 1999a, 2000; Barot & Gignoux, 1999, 2003). The population dynamics of this species heavily interacts with its spatial distribution: seedlings and juveniles do not grow where adults are found (Barot et al., 1999b). This discrepancy suggests that there has been a recent change in seed dispersal patterns, possibly linked to the local extinction of animal dispersers (elephants and baboons). The model we propose here is entirely based on field data and aims at testing the hypothesis that the current dispersal regime cannot maintain the currently observed spatial patterns of adults.

Entities, state variables and scales - To represent the spatial distribution of trees in this model, we used an individual-based model where each individual tree (adult, juvenile, or seedling) is located within a rectangular plot representing a few hectares of savanna. In 3Worlds terms, we have:

- 4 *component types*: *palm seedlings*, *juveniles* and *adults*, and *termite mounds*. Palm types are linked through a *life cycle*;
- 3 relation types used to define 4 different neighbourhood indices, as described in (Barot & Gignoux, 2003);
- 5 driver variables (2 for adults, 2 for juveniles, 1 for seedlings);
- 4 neighbourhood indices as decorators (number of trees, adult palms, juveniles, termite mounds);

- 43 constants, among which the plant locations (x, y) are the most important. Others include the parameters of the regressions found in (Barot et al., 1999a, 2000; Barot & Gignoux, 1999, 2003). See appendix 7 for details;
- the time step is 1 year;
- space is a continuous flat surface of size 300 × 300 m where palm trees and termite mounds are located;
- simulations are run until user intervention.

The dynamic graph at run time comprises hundreds to thousands of *ephemeral* nodes representing individual palm trees of the 3 demographic stages, and *permanent* termite mounds present in the landscape. Palm trees and mounds are linked through *permanent* (life-long) neighbourhood relations.

Process overview and scheduling **–** This model comprises 17 processes with a fairly elaborate organisation (cf. flowchart in appendix 7). First come computations of neighbourhood indices and their consequences:

- *adultNeighbour, juvenileNeighbour, moundNeighbour* (of type *relateToDecision*, fig. 2) respectively relate an adult palm, a juvenile palm and a termite mound, to the neighbours they influence;
- *competitionA*, *competitionJ*, *moundEffect* (of type *relateToDecision*, fig. 2) compute the weight of adults, juveniles and mounds in the neighbourhood indices affecting palm stage biology;

Then come computations describing the biology of every palm stage. Some of them depend on the neighbourhood indices computed before :

- adults:
  - *growAdult* (of type *changeState*, fig. 2) computes the growth of adult palms in height and number of leaves;
  - *mortalityAdult* (of type *deleteDecision*, fig. 2) decides if an adult dies;
  - *reproduction* (of type *createOtherDecision*, fig. 2) computes the number of seedlings produced by an adult female palm;
  - *dispersal* (of type *setOtherInitialState*, fig. 2) computes the initial state of a newborn seedling, essentially its location relative to its mother;
- seedlings:
  - *mortalitySeedling* (of type *deleteDecision*, fig. 2) decides if a seedling dies;
  - *recruitSeedling* (of type *changeCategoryDecision*, fig.2) decides if a seedling mutates to the next stage according to the life cycle, i.e. juvenile;
  - *seedlingToJuvenile* (of type *setOtherInitialState*, fig. 2) carries over internal data from seedling to juvenile (given that the two stages do not have the same descriptors)
- juveniles:

- *growJuvenile* (of type *changeState*, fig. 2) computes the growth of juvenile palms in height and number of leaves;
- *mortalityJuvenile* (of type *deleteDecision*, fig. 2) decides if a juvenile dies;
- *recruitJuvenile* (of type *changeCategoryDecision*, fig.2) decides if a juvenile mutates to the next stage according to the life cycle, i.e. adult;
- *juvenileToAdult* (of type *setOtherInitialState*, fig. 2) carries over internal data from juvenile to adult.

Scheduling and java code for these functions can be found in appendix 7.

Details – Full configuration graph, variable list, flow chart, user code and a screen copy can be found in appendix 7. Model metrics are in Table 2.

Conclusion – This is currently one of the most complex models available in 3Worlds. It uses all the concepts developed in 3Worlds to describe complex life cycles in an individual-based framework, although in a fairly simple case. It should be further expanded by adding the influence of other savanna trees as shown in the references used to parameter the model before running simulation experiments on the influence of dispersal on adult population spatial pattern.

5. Conclusion

Following the principles of *aspect-oriented-thinking* we have developed a flexible system for unambiguously specifying and simulating *ecosystems.*

We have also developed a graph system that can represent any system evolving over time and capture all forms of emergence we have identified.

These two developments underpin 3Worlds: a platform within which any dynamic system can be implemented; where model abstraction level can be manipulated; and in which the causes of different model outcomes can be identified more easily. This is because all models built using 3Worlds will necessarily comply with the one definition of an ecosystem: something that is *multi-aspect*, *scale-independent*, *observer-selected* and *recursive*.

6. Availability

3Worlds code, binary application and documentation are freely available as an Open-source project under the GPL 3.0 license at https://github.com/3worlds/3w.

Currently, 3Worlds comprises a library of 24 models: 10 are test models focusing on a particular feature of the platform, 10 are tutorial models of increasing complexity, 4 are new ecological models of interest to the authors and their collaborators.

It is written in *Java 11* for portability. It totals ~125,000 lines of code organised in 12 libraries (Table 3). It can be run under *linux*, *windows* or *MacOS*. Dependencies are managed using *apache ivy*. Low-level libraries have been tested using *Junit v.5.0*. Higher level libraries have been tested with specific test models available in the distribution. We have made a significant effort to write extensive and useful documentation (~120 pages: https://3worlds.github.io/tw-uifx/tw-uifx/doc/reference/html/reference.html).

Acknowledgements

We thank all the people who contributed their time to the development of the software. We thank Sam Banks, Anne Forsythe, Naoise Nunan and Xavier Raynaud for their reading and editing of the manuscript, and Perrine Cribier-Delande for setting up the github site for public distribution.

Funding

3Worlds is the result of a 20 year collaboration between two teams based in France and Australia. This project would not have been possible without two kinds of funding, unconditional, long-term, informal support, and short-term more substantive support. We thank the Centre national de la recherche scientifique (CNRS, France), the Australian National University (ANU, Australia), and the Charles Darwin University (CDU, Australia) for the former; and the Agence nationale pour la recherche (project ANR-07-CIS7-001-01), the CNRS (international cooperation project DRI N° 16 059), and the Australian research council (grant DP210103227) for the latter.

Bibliography

Achtemeier, G. L. (2003). “Rabbit Rules”–An Application of Stephen Wolfram’s “New Kind of Science” to Fire Spread Modeling. *Fifth Symposium on Fire and Forest Meteorology*, 16–20. https://ams.confex.com/ams/pdfpapers/65944.pdf

Achtemeier, G. L. (2013). Field validation of a free-agent cellular automata model of fire spread with fire—Atmosphere coupling. *International Journal of Wildland Fire*, *22*(2), 148. https://doi.org/10.1071/WF11055

Amouroux, E., Chu, T. Q., Boucher, A., & Drogoul, A. (2009). GAMA: An Environment for Implementing and Running Spatially Explicit Multi-agent Simulations. *Agent Computing and Multi-Agent Systems*, *5044*, 359–371.

Barot, S., & Gignoux, J. (1999). Population structure and life cycle of *Borassus aethiopum* Mart.: Evidence of senescence in a palm tree. *Biotropica*, *31*(3), 439–448.

Barot, S., & Gignoux, J. (2003). Neighbourhood analysis in the savanna palm *Borassus aethiopum*: Interplay of intraspecific competition and soil patchiness. *Journal of Vegetation Science*, *14*(1), 79–88.

Barot, S., Gignoux, J., & Menaut, J. (1999a). Seed shadows, survival and recruitment: How simple mechanisms lead to the dynamics of population recruitment curves. *Oikos*, *86*, 320–330.

Barot, S., Gignoux, J., & Menaut, J. C. (1999b). Demography of a savanna palm tree: Predictions from comprehensive spatial pattern analyses. *Ecology*, *80*(6), 1987–2005.

Barot, S., Gignoux, J., Vuattoux, R., & Legendre, S. (2000). Demography of a savanna palm tree in Ivory Coast (Lamto): Population persistence, and life history. *Journal of Tropical Ecology*, *16*, 637–655.

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. *Software: Practice and Experience*, *31*(2), 103–128. https://doi.org/10.1002/1097-024X(200102)31:2<103::AID-SPE358>3.0.CO;2-O

Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P., & Slack, M. G. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, *9*(2–3), 237–256. https://doi.org/10.1080/095281397147103

Bousquet, F., & Le Page, C. (2004). Multi-agent simulations and ecosystem management: A review. *Ecological Modelling*, *176*(3–4), 313–332. https://doi.org/10.1016/j.ecolmodel.2004.01.011

Bugmann, H. K. M., Yan, X. D., Sykes, M. T., Martin, P., Lindner, M., Desanker, P. V., & Cumming, S. G. (1996). A comparison of forest gap models: Model structure and behaviour. *Climatic Change*, *34*(2), 289–313.

Cary, G. J., Keane, R. E., Gardner, R. H., Lavorel, S., Flannigan, M. D., Davies, I. D., Li, C., Lenihan, J. M., Rupp, T. S., & Mouillot, F. (2006). Comparison of the Sensitivity of Landscape-fire-succession Models to Variation in Terrain, Fuel Pattern, Climate and Weather. *Landscape Ecology*, *21*(1), 121–137. https://doi.org/10.1007/s10980-005-7302-9

Chow, A. C., Zeigler, B. P., & Doo Hwan Kim. (1994). Abstract simulator for the parallel DEVS formalism. *Fifth Annual Conference on AI, and Planning in High Autonomy Systems*, 157–163. https://doi.org/10.1109/AIHAS.1994.390488

Connell, J. H. (1978). Diversity in tropical rain forests and coral reefs — High diversity of trees and corals is maintained only in a nonequilibrium state. *Science*, *199*(4335), 1302–1310.

Davies, I. D. (2014). *Scale and Abstraction: The Sensitivity of Fire-Regime Simulation to Nuisance Parameters.* PhD, Australian National University.

Dorri, A., Kanhere, S. S., & Jurdak, R. (2018). Multi-Agent Systems: A Survey. *IEEE Access*, *6*, 28573–28593. https://doi.org/10.1109/ACCESS.2018.2831228

Ermentrout, G. B., & Edelstein-Keshet, L. (1993). Cellular Automata Approaches to Biological Modelling. *Journal of Theoretical Biology*, *160*, 97–133.

Favier, C., & Dubois, M. A. (2004). Reconstructing forest savanna dynamics in Africa using a cellular automata model, FORSAT. *Lecture Notes in Computer Science*, *3305*, 484–491.

Ferber, J. (1995). *Les systèmes multi-agents. Vers une intelligence collective*. InterEditions.

Flint, S. R. (2006). *Aspect-Oriented Thinking—An approach to bridging the disciplinary divides.* PhD, Australian National University.

Friedlingstein, P., Cox, P., Betts, R., Bopp, L., Von Bloh, W., Brovkin, V., Cadule, P., Doney, S., Eby, M., Fung, I., & others. (2006). Climate-carbon cycle feedback analysis: Results from the C4MIP model intercomparison. *Journal of Climate*, *19*(14), 3337–3353.

Gauzens, B., Legendre, S., Lazzaro, X., & Lacroix, G. (2013). Food-web aggregation, methodological and functional issues. *Oikos*, *122*(11), 1606–1615. https://doi.org/10.1111/j.1600-0706.2013.00266.x

Gignoux, J., Chérel, G., Davies, I. D., Flint, S. R., & Lateltin, E. (2017). Emergence and complex systems: The contribution of dynamic graph theory. *Ecological Complexity*, *31*, 34–49. https://doi.org/10.1016/j.ecocom.2017.02.006

Gignoux, J., Davies, I. D., Flint, S. R., & Zucker, J.-D. (2011). The Ecosystem in Practice: Interest and Problems of an Old Definition for Constructing Ecological Models. *Ecosystems*, *14*(7), 1039–1054. https://doi.org/10.1007/s10021-011-9466-2

Gignoux, J., Menaut, J. C., Noble, I. R., & Davies, I. D. (1998). A spatial model of savanna dynamics: Model description and preliminary results. In D. M. Newbery, H. H. T. Prins, & N. D. Brown (Eds.), *Dynamics of tropical communities* (pp. 361–383). Blackwell Science.

Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S. K., Huse, G., Huth, A., Jepsen, J. U., Jørgensen, C., Mooij, W. M., Müller, B., Pe'er, G., Piou, C., Railsback, S. F., Robbins, A. M., … DeAngelis, D. L. (2006). A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, *198*(1–2), 115–126. https://doi.org/10.1016/j.ecolmodel.2006.04.023

Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., & Railsback, S. F. (2010). The ODD protocol: A review and first update. *Ecological Modelling*, *221*(23), 2760–2768. https://doi.org/10.1016/j.ecolmodel.2010.08.019

Grimm, V., & Railsback, S. (2005). *Individual-based modelling and ecology*. Princeton University Press.

Gritti, E. S., Gaucherel, C., Crespo-Perez, M.-V., & Chuine, I. (2013). How Can Model Comparison Help Improving Species Distribution Models? *PLoS ONE*, *8*(7), e68823. https://doi.org/10.1371/journal.pone.0068823

Gross, J. L., & Yellen, J. (1999). *Graph Theory and Its Applications*. Chapman & Hall/CRC.

Gurney, W. S. C., & Nisbet, R. M. (1998). *Ecological Dynamics*. Oxford University Press.

Hannay, J. E., MacLeod, C., Singer, J., Langtangen, H. P., Pfahl, D., & Wilson, G. (2009). How do scientists develop and use scientific software? *2009 ICSE Workshop on Software*

Engineering for Computational Science and Engineering, 1–8. https://doi.org/10.1109/SECSE.2009.5069155

Hantson, S., Kelley, D. I., Arneth, A., Harrison, S. P., Archibald, S., Bachelet, D., Forrest, M., Hickler, T., Lasslop, G., Li, F., Mangeon, S., Melton, J. R., Nieradzik, L., Rabin, S. S., Prentice, I. C., Sheehan, T., Sitch, S., Teckentrup, L., Voulgarakis, A., & Yue, C. (2020). Quantitative assessment of fire and vegetation properties in simulations with fire-enabled vegetation models from the Fire Model Intercomparison Project. *Geoscientific Model Development*, *13*(7), 3299–3318. https://doi.org/10.5194/gmd-13-3299-2020

Harary, F., & Gupta, G. (1997). Dynamic graph models. *Mathematical and Computer Modelling*, *25*(7), 79–87. https://doi.org/10.1016/S0895-7177(97)00050-2

Hernandez Encinas, A., Hernandez Encinas, L., Hoya White, S., Martin del Rey, A., & Rodriguez Sanchez, G. (2007). Simulation of forest fire fronts using cellular automata. *Advances in Engineering Software*, *38*(6), 372–378.

Intergovernmental Panel on Climate Change (Ed.). (2014). Evaluation of Climate Models. In *Climate Change 2013 – The Physical Science Basis: Working Group I Contribution to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change.* (pp. 741–866). Cambridge University Press; Cambridge Core. https://doi.org/10.1017/CBO9781107415324.020

Jepsen, J. U., Baveco, J. M., Topping, C. J., Verboom, J., & Vos, C. C. (2005). Evaluating the effect of corridors and landscape heterogeneity on dispersal probability: A comparison of three spatially explicit modelling approaches. *Ecological Modelling*, *181*(4), 445–459. https://doi.org/10.1016/j.ecolmodel.2003.11.019

Keane, R. E., Cary, G. J., Davies, I. D., Flannigan, M. D., Gardner, R. H., Lavorel, S., Lenihan, J. M., Li, C., & Rupp, T. S. (2004). A classification of landscape fire succession models: Spatial simulations of fire and vegetation dynamics. *Ecological Modelling*, *179*(1), 3–27. https://doi.org/10.1016/j.ecolmodel.2004.03.015

Kleijnen, J. P. C., Sanchez, S. M., Lucas, T. W., & Cioppa, T. M. (2005). State-of-the-Art Review: A User's Guide to the Brave New World of Designing Simulation Experiments. *INFORMS Journal on Computing*, *17*(3), 263–289. https://doi.org/10.1287/ijoc.1050.0136

Lenhard, J., & Winsberg, E. (2010). Holism, entrenchment, and the future of climate model pluralism. *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics*, *41*(3), 253–262. https://doi.org/10.1016/j.shpsb.2010.07.001

Lim, W. H., & Roderick, M. L. (2009). *An atlas of the global water cycle based on the IPCC AR4 climate models*. ANU E Press.

Melilo, J. M., Borchers, J., Chaney, J., Fisher, H., Fox, S., Haxeltine, A., Janetos, A., Kicklighter, D. C., Kittel, T. G. F., McGuire, A. D., McKeown, R., Neilson, R., Nemani, R., Ojima, D. S., Painter, T., Pan, Y., Parton, W. J., Pierce, L., Pitelka, L., … Woodward, F. I. (1995). Vegetation/ecosystem modeling and analysis project: Comparing biogeography and biogeochemistry models in a continental-scale study of terrestrial ecosystem responses to climate change and CO2 doubling. *Global Biogeochemical Cycles*, *9*(4), 407–437.

Minar, N., Burkhart, R., Langton, C., & Askenazi, M. (1996). *The swarm simulation system: A toolkit for building multi-agent simulations.* Working Paper No. 96-06–042; 11 pp., Santa Fe Institute. http://cobweb.cs.uga.edu/~maria/pads/papers/swarm-MinarEtAl96.pdf

Muci, A. L., Jorquera, M. A., Avila, A. I., Rengel, Z., Crowley, D. E., & Mora, M. D. (2012). A combination of cellular automata and agent-based models for simulating the root surface colonization by bacteria. *Ecological Modelling*, *247*, 1–10.

North, M. J., Tatara, E., Collier, N. T., & Ozik, J. (2007). Visual Agent-based Model Development with Repast Simphony. *Proceedings of the Agent 2007 Conference on Complex Interaction and Social Emergence*.

Peck, S. L. (2004). Simulation as experiment: A philosophical reassessment for biological modeling. *Trends in Ecology & Evolution*, *19*(10), 530–534. https://doi.org/10.1016/j.tree.2004.07.019

Piela, P. C., Epperly, T. G., Westerberg, K. M., & Westerberg, A. W. (1991). ASCEND: An object-oriented computer environment for modeling and analysis: The modeling language. *Computers & Chemical Engineering*, *15*, 53–72.

Quesnel, G., Duboz, R., & Ramat, E. (2009). The Virtual Laboratory Environment—An operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Practice and Theory*, *17*(4), 641–653.

Reuillon, R., Leclaire, M., & Rey-Coyrehourcq, S. (2013). OpenMOLE, a workflow engine specifically tailored for the distributed exploration of simulation models. *Future Generation Computer Systems*, *29*(8), 1981–1990. https://doi.org/10.1016/j.future.2013.05.003

Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, *21*(4), 25–34.

Reynolds, C. W. (1999). *Steering Behaviors For Autonomous Characters*. 21pp.

Richmond, B., Peterson, S., & Vescuso, P. (1987). *An Academic User's Guide to STELLA*. High Performance Systems, Inc.

Roxburgh, S. H., Barrett, D. J., Berry, S. L., Carter, J. O., Davies, I. D., Gifford, R. M., Kirschbaum, M. U. E., McBeth, B. P., Noble, I. R., Parton, W. G., Raupach, M. R., & Roderick, M. L. (2004). A critical overview of model estimates of net primary productivity for the Australian continent. *Functional Plant Biology*, *31*(11), 1043–1059.

Samet, H. (1984). The Quadtree and Related Hierarchical Data Structures. *Computing Surveys*, *16*, 187–260.

Smith, J., Smith, P., & Addiscott, T. (1996). Quantitative methods to evaluate and compare soil organic matter (SOM) models. In D. Powlson, P. Smith, & J. Smith (Eds.), *Evaluation of soil organic matter models* (Vol. 38, pp. 181–200). Springer Verlag, Berlin.

Smith, P., Smith, J., Powlson, D., McGill, W., Arah, J., Chertov, O., Coleman, K., Franko, U., Frolking, S., Jenkinson, D., Jensen, L., Kelly, R., Klein-Gunnewiek, H., Komarov, A., Molina, J., Mueller, T., Parton, W., Thornley, J., & Whitmore, A. (1997). A comparison of the performance of nine soil organic matter models using datasets from seven long-term experiments. *Geoderma*, *81*, 153–225.

Steinhagen, R. J., Bräuning, H., Krimm, A., & Milosic, T. (2019). Redesign of the JavaFX Charts Library in View of Real-Time Visualisation of Scientific Data. *Proc. 10th International Particle Accelerator Conference (IPAC'19), Melbourne, Australia, 19-24 May 2019*, 3868–3871. https://doi.org/doi:10.18429/JACoW-IPAC2019-THPRB028

Tansley, A. G. (1935). The use and abuse of vegetational concepts and terms. *Ecology*, *16*, 284–307.

Thom, R., & Noël, E. (1993). *Prédire n'est pas expliquer*. Flammarion.

Wilensky, U. (1999). *NetLogo: Http://ccl.northwestern.edu/netlogo/*.

Woodcock, J., Larsen, P. G., Bicarregui, J., & Fitzgerald, J. (2009). Formal methods: Practice and experience. *ACM Computing Surveys*, *41*(4), 9.1-9.36. https://doi.org/10.1145/1592434.1592436

Zeigler, B. P., Moon, Y., Kim, D., & Ball, G. (1997). The DEVS environment for high-performance modeling and simulation. *IEEE Computational Science & Engineering*, *4*(3), 61–71. https://doi.org/10.1109/99.615432

Zucker, J.-D. (2003). A grounded theory of abstraction in artificial intelligence. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *358*(1435), 1293–1309. https://doi.org/10.1098/rstb.2003.1308

Tables

border type	***effect on edge***
wrap	objects crossing the border reappear at the opposite end of that same dimension. Therefore these borders must come as a pair and cannot be applied to circles or spheres for example.
reflection	a hard border at which objects bounce
sticky	a hard border to which objects stick
oblivion	objects crossing the border are removed from the simulation
infinite	no border – dimension extends in one direction following object movements
border combination	***effect on space***
periodic	*Symmetric, finite and unbounded: wrap* in all dimensions, (e.g. topologically a torus in a 2 dimensional space)
reflective	*Symmetric, finite and bounded*: all borders are reflective, i.e. objects bounce at borders
island	*Symmetric, finite and bounded*: all borders are oblivious, i.e. items crossing any border are lost from the simulation
unbounded	*Symmetric, infinite and unbounded*: an infinite space adapting to location of objects (all borders are *infinite*)
bounded	*Symmetric, finite and bounded*: a space with sticky borders in all directions, i.e. objects that arrive at the border stay there forever
tubular	*Asymmetric, finite and unbounded in one dimension, bounded in all others*: wrap around borders in the first dimension, sticky borders in all other dimensions
custom	user-specified border properties – provide a (possibly different) ***border type*** property for each side of the space

Table 1. Edge effect-correction methods. Whatever its dimension, the simulated space has borders; the *border type* property defines how a spatial object located in the space interacts with a single border. *Border combination*s are a few predefined standard settings of space commonly used, but total freedom is left to define other combinations (*custom*). Finally, an *observation window* smaller than the space can be defined: objects can exist anywhere in the space, but only outputs from within the observation window are reported.

model	*configuration size*	*descriptors*			*classifiers*		*complexity*
		drv	*cst*	*dec*	*component types*	*relation types*	*CCUCS (bytes)*
RNG	40	2	0	0	0	0	2,988
spatial	51	2	0	0	1	0	3,356
logistic	33	1	1	0	0	0	4,498
pulseNS	69	0	3	0	1	0	4,588
lotkavolterra	57	4	12	0	0	0	5,557
pulseS	105	2	3	0	1	0	9,012
IDH	136	41	122	1	2	1	14,060
panmixia	126	2	6	0	1	1	14,909
boids	189	6	5	9	1	1	16,301
Rabbit Rules	194	270,011	90,019	0	0	0	17,638
littleForest	150	1	6	1	1	1	18,353
LMA	245	10,003	8	30,006	1	0	31,148
LMB	321	10,005	12	40,026	1	0	37,404
LMC	384	10,005	15	40,025	1	1	57,752
palms	656	5	43	5	4	3	60,109
LMD	519	10,005	26	207,864	1	1	75,637

Table 2. Some measures of model size and complexity for models developed in 3Worlds so far. *configuration size* = number of nodes + edges + properties of the model configuration graph; *drv* = drivers, or state variables; *cst* = constants; *dec* = decorators, or secondary variables; classifiers = numbers of component and relation types of different categories; CCUCS = the size of the compressed compiled user code in bytes. If we assume the user code is efficiently written and compiled, then we can consider its size as a measure of Kolmogorov complexity (Kolmogorov, 1963, cited by wikipedia) of the ecosystem it represents. Very large numbers of descriptors indicate the model uses tables: each table cell is counted as one descriptor.

library	***content***	***depends on***
omhtk	generic concepts and utilities	*org.apache.commons:commons-io*
omugi	lightweight graph implementation	omhtk
rvgrid	Ada rendezvous + generic state machine implementations	omhtk
uit	generic K-d trees	omhtk
qgraph	tools for searching & querying graphs	omhtk, omugi
ymuit	Javafx utilities for user interface	omhtk, uit, *org.openjfx:javafx-controls, org.openjfx:javafx-graphics, org.openjfx:javafx-base*
tw-models	library of 3worlds models	omugi
aot	aspect-oriented thinking tools	omhtk, omugi, qgraph
tw-core	the core of 3Worlds	omhtk, uit, rvgrid, omugi, qgraph, aot, *org.apache.commons:commons-math3, org.apache.commons:commons-text, org.apache.odftoolkit:simple-odf, com.hp.hpl.jena:jena*
tw-setup	packaging utility	omhtk, tw-core, *org.apache.ivy:ivy*
tw-apps	**ModelMaker** & **ModelRunner** code	omhtk, omugi, qgraph, aot, tw-core, tw-models, *org.apache.commons:commons-text*
tw-uifx	Javafx implementation of **ModelMaker**, **ModelRunner** and the 3Worlds graphical user interface collection of simulation output ’widgets’.	omhtk, rvgrid, omugi, ymuit, tw-models, qgraph, aot, tw-core, tw-apps, *org.openjfx:javafx-fxml, org.controlsfx:controlsfx, de.gsi:chartfx, de.gsi:chartfx-samples, de.gsi.chart:chartfx-chart, de.gsi.dataset:chartfx-dataset, de.gsi.math:chartfx-math, de.gsi.acc:chartfx-acc, de.gsi:microservice, org.slf4j:slf4j-api, org.openjfx:javafx-controls, org.openjfx:javafx-graphics, org.openjfx:javafx-base, org.apache.commons:commons-math3*

Table 3. 3Worlds libraries and their dependencies (external dependencies in *italics*). All 3Worlds libraries can be downloaded from **https://github.com/3worlds/<*library*>**

Figure legends

Fig. 1. The (simplified) configuration tree of **ModelMaker**, as expected from the 3Worlds specification archetype. All models developed in 3Worlds must declare nodes compatible with this general requirement.

Fig. 2. Illustration of the dynamic graph used to represent ecosystems in 3Worlds. Circles = nodes (different colours represent membership to different categories); green lines = edges; blue lines = changes. Top: a dynamic graph with components c_i and relations r_j undergoes some transformations along five successive time steps. Transformations are local and propagate along the graph structure (arrows). Bottom: illustration of all the possible transformations that can occur to a component or a relation; ecological *processes* must map to these transformations. Some of them have direct interpretation (i.e. `createOther` = reproduction, `deleteDecision` = death, etc...).

Fig. 3. UML class diagram showing the central role of the category concept in 3Worlds. For any element of the system (i.e. any node of the system graph), its descriptors and the processes that can act on it are determined by categories and relations.

Fig. 4. Example of a category tree. `CategorySets` are in gray and `Categories` are in green. Categories within the same set are mutually exclusive, ie a component is either `plant` or `animal`, `biotic` or `abiotic`, `tree` or `grass`, etc. In this example, a component can be '`abiotic:fire`' or '`biotic:juvenile:animal`' or '`biotic:senescent:plant:tree:C3`'.

Figures

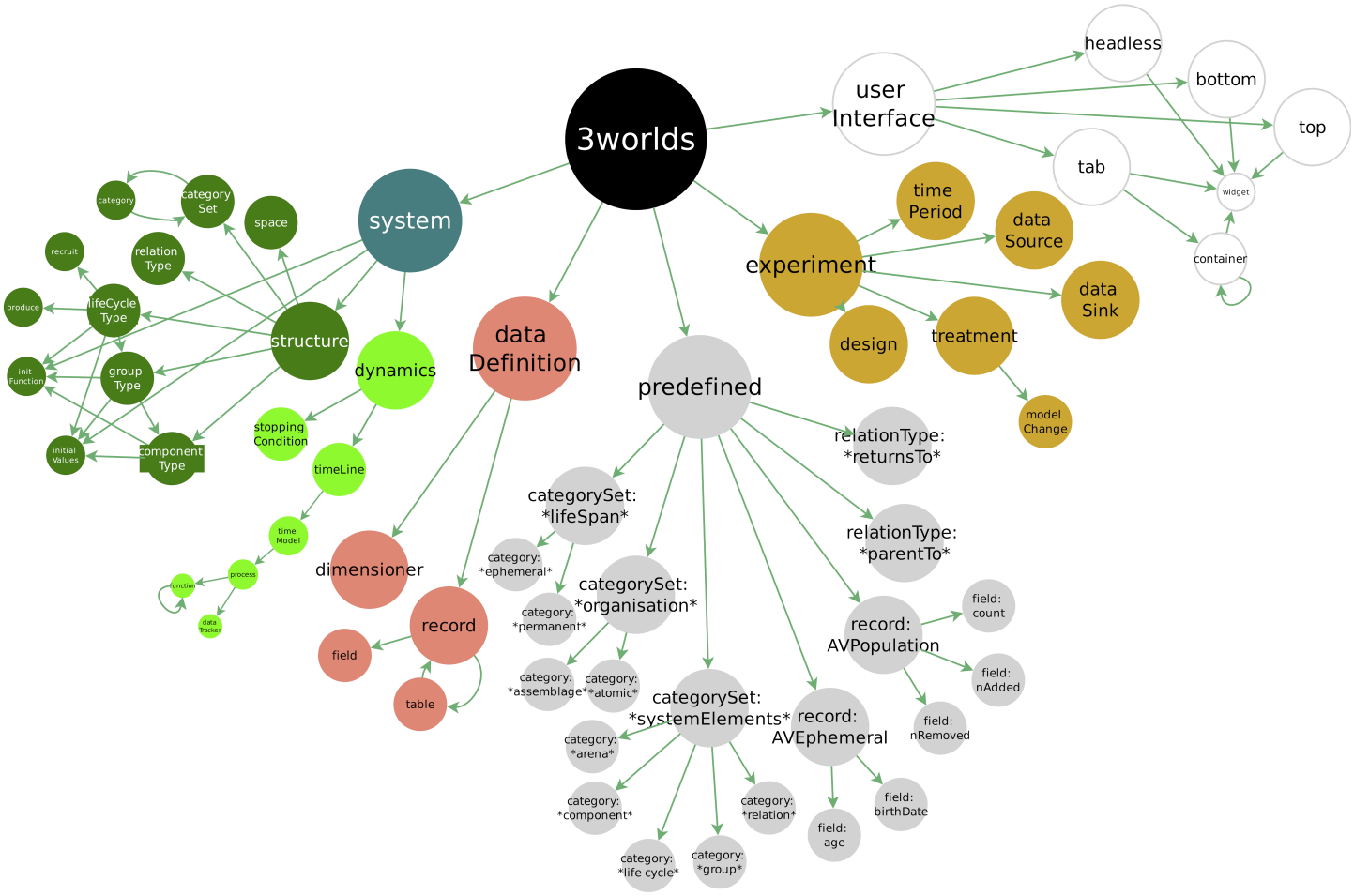

Figure 1

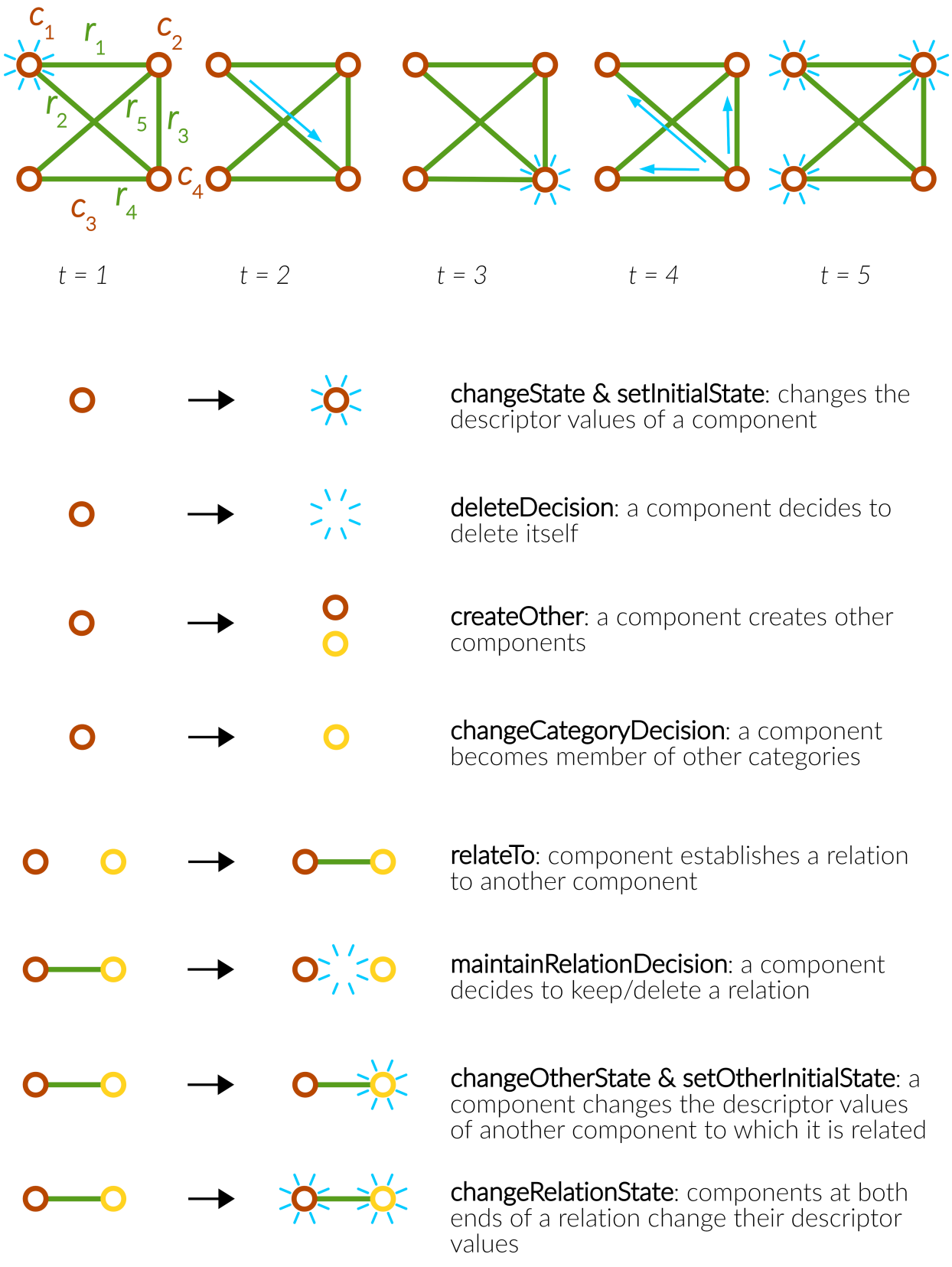
c1
r1
c2
r2
r5
r3
c4
c3
r4
t = 1
t = 2
t = 3
t = 4
t = 5
changeState & setInitialState: changes the descriptor values of a component
deleteDecision: a component decides to delete itself
createOther: a component creates other components
changeCategoryDecision: a component becomes member of other categories
relateTo: component establishes a relation to another component
maintainRelationDecision: a component decides to keep/delete a relation
changeOtherState & setOtherInitialState: a component changes the descriptor values of another component to which it is related
changeRelationState: components at both ends of a relation change their descriptor values

Figure 2

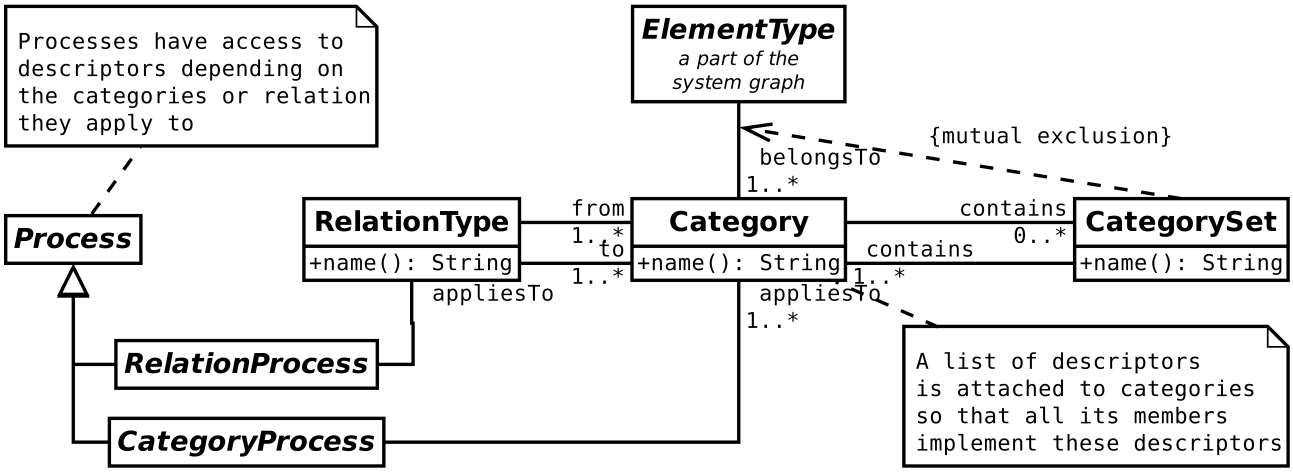

Figure 3

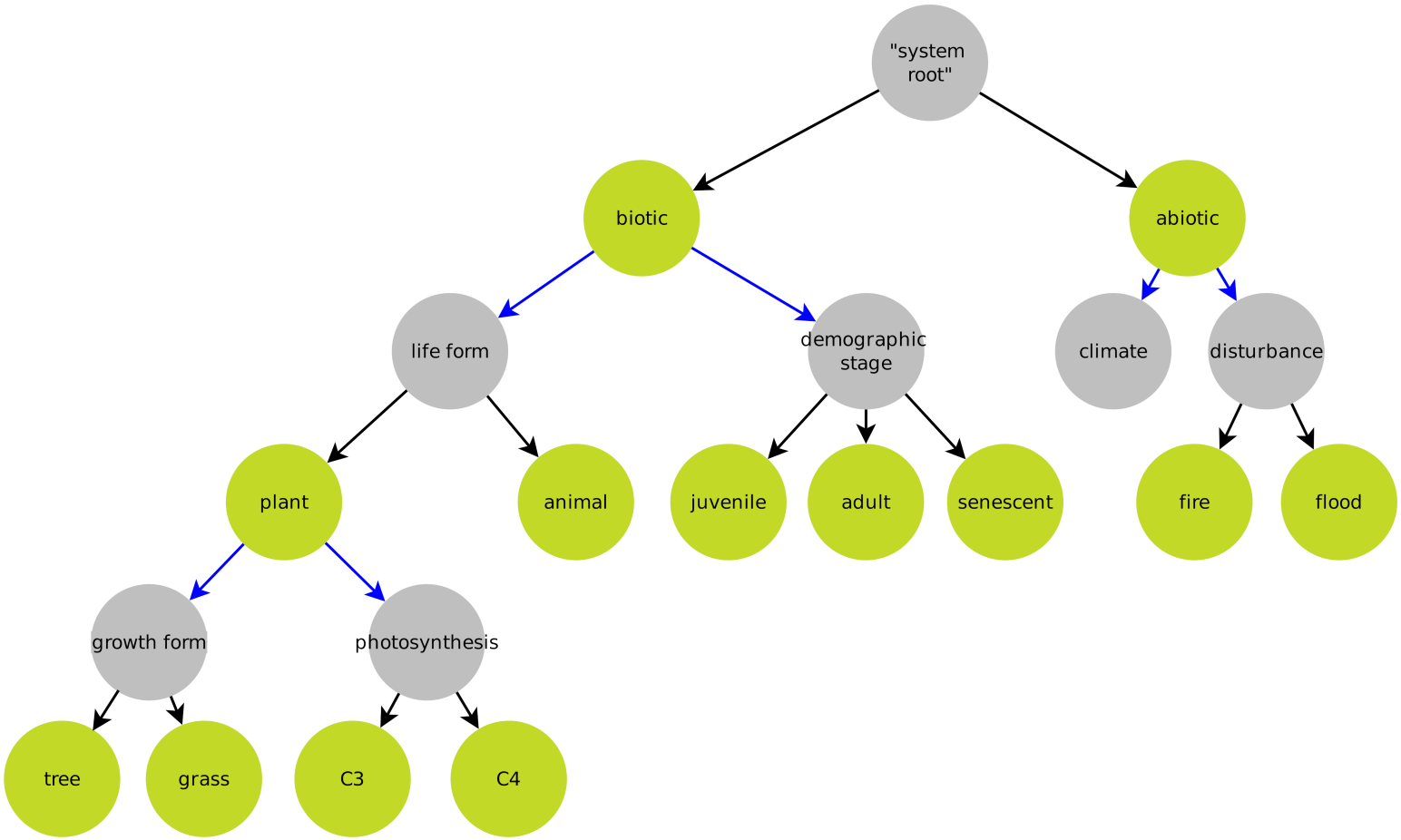

Figure 4