Developing multidisciplinary mechanistic models: challenges and approaches

Daniel Vedder[1,2,3], Samuel M. Fischer[4], Kerstin Wiegand[5,6], Guy Pe'er[1,3]

19th March 2024

[1] Department of Biodiversity and People, Helmholtz Center for Environmental Research - UFZ, Leipzig
[2] Institute of Biodiversity, Friedrich Schiller University Jena
[3] German Center for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig
[4] Department of Ecological Modelling, Helmholtz Center for Environmental Research - UFZ, Leipzig
[5] Department of Ecosystem Modelling, University of Göttingen, Germany
[6] Centre of Biodiversity and Sustainable Land Use (CBL), University of Göttingen, Germany
Corresponding author: Daniel Vedder, daniel.vedder@idiv.de
Puschstraße 4, 04103 Leipzig, Germany

1. Mechanistic models are a valuable tool for understanding and forecasting the dynamics of ecosystems in the face of global crises, such as biodiversity loss and climate change. However, current biodiversity models often struggle to adequately represent the complexity of the involved systems, as these are shaped by many different ecological, physical, and social processes. Consequently, we will need larger and more complex ecological models. Additionally, we will have to couple ecological models to models produced by other disciplines, such as climate science, economics, or sociology. Constructing such integrated models is a significant technical undertaking, which has received little attention by ecological modellers so far.
2. We review literature from computer science and several other environmental modelling disciplines to identify technical challenges with and approaches to creating large integrated models. We show that there is a software-architectural trade-off between modularity and integration, where the former is required to keep the technical complexity of a model manageable, and the latter is desirable to represent

the scientific complexity of a studied system. We then present and compare five different software engineering techniques for navigating this trade-off.

3. Which technique is most suitable for a given model depends on the model's aims and the available development resources. Simple models can be built monolithically, but larger models need to be subdivided into modules or packages to remain manageable. For very large or interdisciplinary models, model coupling on an ad-hoc basis or with a model coupling framework enables modellers to address problems spanning multiple domains or scales.

4. Other modelling disciplines have developed useful techniques for creating large integrated models. Ecological modellers need to learn from these disciplines and invest in increased software engineering expertise, if they want to build biodiversity models that are capable of jointly representing the numerous processes affecting ecosystems and biodiversity loss.

Keywords: ecological modelling, mechanistic models, model complexity, model coupling, FAIR principles, research software

1 Introduction

Mechanistic models (also known as process-based models) have established themselves as an important pillar of ecological research (Pilowsky et al., 2022). They help us better understand ecological processes and patterns (DeAngelis & Grimm, 2014), and are increasingly widely used to study the causes and effects of biodiversity loss (IPBES, 2016). In this, they show great potential for making ecology a more predictive science (McIntire et al., 2022; Stillman et al., 2016), as well as for supporting decision making (Grimm, Johnston et al., 2020; Will et al., 2021).

Mechanistic models are already used to model the full spectrum of biological scales, from individual-level genetic (Romero-Mujalli et al., 2019) and physiological processes (Sibly et al., 2013) all the way up to macroevolutionary processes that shape global patterns (Cabral et al., 2017). However, most ecological models only consider a very small number of processes (Urban et al., 2016), partly because of the complexity of ecosystems and the multitude of processes that need to be modelled. This is problematic, as gaining a full understanding of the natural world will require models that integrate multiple processes and organisational levels, including their interactions across scales (Grimm et al., 2017; Urban et al., 2022).

In addition to integrating the various strands of ecological research, mechanistic models can be used to link ecology to other scientific disciplines. This is vital in the context of complex issues such as Global Change, where we need to understand how ecosystems and biodiversity affect and are affected by both physical domains such as climate (Urban et al., 2016) and socio-economic domains such as agriculture (Malawska et al., 2014). Ultimately, if we want to gain a deep, holistic understanding of the natural world we live in, we will not just need more comprehensive ecological models, but also integrated models that create a link between ecology and the physical and social sciences (Cabral et al., 2023).

Building such larger and more integrated models entails high scientific complexity, and there is a lively discussion among ecological modellers about whether and when this is necessary and how to deal with this (e.g. Lorscheid & Meyer, 2016; Sun et al., 2016; Topping et al., 2015). However, there is comparatively little discussion about the technical aspects of constructing large models. Models are software systems that also have a high technical complexity; thus, creating large integrated models will mean building large complicated software (Johanson & Hasselbring, 2018; Sanders & Kelly, 2008; Vedder, Ankenbrand & Cabral, 2021).

Of course, there are multiple dimensions in which model software can be improved, including increased realism and computational efficiency. However, with the growing importance of interdisciplinary, and especially social-ecological, research, the aim of combining models from multiple domains has taken on new urgency (Vedder et al., submitted). While we should continue to think about how to build better monodisciplinary models ("go deep"), we should at the same time strive to build more multidisciplinary models ("go wide").

Therefore, with this review, we address the question of how such model integration can be practically achieved. We juxtapose relevant principles from the computer science literature on software engineering with examples from current practice in ecological modelling. To help link the two, we also draw on the wider literature on scientific computing, and include examples from other modelling fields such as climate modelling. We identify the challenges in creating large integrated models, and discuss the advantages and disadvantages of different techniques that can be used to build them. We end by drawing together concrete and practical advice for creating large integrated models.

Box 1: Glossary of software terms

Clean code Computer code that is easy to read, understand, and modify (see Martin, 2009, for an in-depth discussion).

Code base The complete set of source code files for a software application, often taken to include the associated input files and documentation.

Complexity ceiling The maximum attainable complexity of a →code base. As the size and technical complexity of a code base increases, the difficulty and cost of adding new features and fixing bugs increases further and further, partly due to accumulating →technical debt. Eventually, further development becomes unfeasible and the software must be rewritten or left as-is. Good →software engineering and →clean code can raise the complexity ceiling (cf. Martin, 2009).

Model coupling The joining of two or more models, so that the output of one is used as input for another. May be accomplished using data files, software packages, network connections, or coupling frameworks (see sections 3.3 to 3.5, and the overview in Belete et al., 2017).

Modularity When referring to the internal structure of a →code base: the property of being subdivided into semi-independent modules, or the degree of subdivision. Structuring software into self-contained modules makes it easier to understand and modify, as most changes will only affect a small subset of the entire code. (See Abelson et al., 1996, and McConnell, 2004, for detailed discussions.)

Software architecture The internal structure of a →code base, including it's organisation into packages, files, classes, and/or functions. Designing this is an important part of →software engineering.

Software engineering The science and practice of developing software applications. The term emphasises the planning, design, and quality control procedures required for implementing large software projects (cf. McConnell, 2004). This is in contrast to the terms "programming" or "software development", which often focus more on the actual act of writing code.

Technical debt Unnecessary technical complexity in a →code base that makes it difficult to understand and work with. This may be caused by a →software architecture that is either too simple or too complicated, a disregard of the principles of →clean code, or incomplete documentation. Although writing such suboptimal code may be faster at first, the resulting (unnecessarily high) technical complexity makes future development harder and slower unless the software quality is improved (i.e. the debt is "paid back").

2 The trade-off between modularity and integration

Large, interdisciplinary models are challenging to build both scientifically and technically (Vedder, Ankenbrand & Cabral, 2021). The larger model software becomes, the more its creation must be treated as software engineering and not merely as software development (Johanson & Hasselbring, 2018). Engineering large software systems is hard, and computer scientists have been thinking for decades about how to deal with their inherent and unavoidable complexity (see e.g. the seminal papers by Brooks, 1986; Dijkstra, 1972).

A fundamental solution to address this complexity is the concept of modularity, and the related concepts of encapsulation and abstraction (e.g. Abelson et al., 1996; McConnell, 2004). Complex software systems are easier to develop and understand when they are split into semi-independent subsystems. Two complementary aspects of modularity are "low coupling" and "information hiding" (Beck & Diehl, 2011). Low coupling means that each subsystem should be as self-contained as possible, with few links to the rest of the program. This makes it easier to develop, test, and analyse each subsystem in isolation. Information hiding means that each subsystem should have a defined and restricted interface through which the rest of the program can access its functionality without having to know the implementation details. This makes it possible to treat each subsystem as a "black box", reducing the complexity of the complete system and making it much easier to think about and design. Additionally, it means that subsystems are easier to replace if and when that is necessary. Thus, the concept of modularity is directly applicable to the development of mechanistic models in ecology, and is indeed used in a similar manner to deal with scientific complexity (Lorscheid & Meyer, 2016).

However, the technical aim of keeping subsystems independent can come into conflict with the scientific desire to represent the numerous interlinkages between domains in the real world (Lippe et al., 2019; Topping et al., 2015). Many ecological models include interactions between processes at different scales (e.g. climate change and animal breeding behaviour), or between different entities at the same scale (e.g. predator and prey species). This requires the model source code to be sufficiently integrated to allow these interactions to be represented. On a technical level, the degree of integration of a software can be measured as the number of linkages between its components (e.g. the number of functions or variables in one component that are referenced outside of this component). The greater the number of linkages, the higher the integration and the lower the modularity (Beck & Diehl, 2011).

Therefore, there is a scientific desire for greater integration to increase realism and a technical desire for greater modularity to increase code tractability, leading to a trade-off between these two model software properties. Modellers must be aware of this trade-off, and carefully weigh the scientific benefits of increased integration against its associated technical complexity costs. We therefore recommend the principle of writing model code that is "as modular as possible, as integrated as necessary".

3 Techniques for creating integrated models

Below, we examine practical approaches for reconciling the scientific need for model integration with the technical imperative of software modularity. For this discussion, we use the terms "model" and "component" in a very specific sense. By "model" we mean a stand-alone executable software that is designed to represent one or more aspects of the physical world. By "component" we mean a subsystem of such a model, that handles one aspect of the model's purpose and interacts with the model's other components. Such a component may be a software module, package, or library, or even another model. Consequently, a stand-alone model may also be used as a component when it is coupled together with other components to create an integrated model. The definitions of model and component in this context are therefore functional and not mutually exclusive: a model is executable, a component interoperable.

Multiple techniques can be used to create large integrated models, each with their own benefits and drawbacks (Brandmeyer & Karimi, 2000). Fundamentally, the important questions are how to implement and link components. As complex software usually involves work by multiple developers or even teams of developers, the tasks of creating (and extending or adapting) components and models will often be done by different people and sometimes by different groups. It is therefore important to consider both the perspectives of component developers and model developers.

In this section we will present five techniques, which represent increasingly advanced types of software architectures that allow progressively greater levels of complexity to be handled (Fig. 1a). The techniques are characterised by differing degrees of interdependence between the model and its components (Fig. 1b). Our selection of techniques is not meant to be comprehensive, but rather aims (1) to show the breadth of available options, (2) give examples of how they have been used in ecological modelling, and (3) discuss important differences between them. We do not propose that any of these techniques are always better or worse than the others; instead, we examine how well they suit specific research questions or organisational contexts. In this analysis, we focus on the seven aspects listed and summarised in Fig. 2.

3.1 Monolithic models

The simplest way to implement a mechanistic model is as a monolithic, purpose-built software that encompasses exactly those entities and processes relevant to the study question (Grimm & Railsback, 2005). Focusing on a specific question allows the developers to reduce complexity by excluding anything that is not directly necessary. At the same time, every part of the model can easily interact with every other part, allowing full integration. Hence, there is a low engineering overhead to the initial model construction. It is likely that the great majority of ecological models fall into this category, as modellers typically create new models for their specific needs and rarely share code (Bell et al., 2015; Berger et al., 2024).

The problem with this design is that, because every part of the code can influence every other part, the complexity of the code can end up increasing exponentially as new features are added. Eventually, the project may experience a "complexity ceiling" effect, where the code base has become so complicated that the developers struggle to expand it further without breaking existing functionality (Martin, 2009). This is why computer science developed modular programming techniques such as object-oriented programming, as these subdivide and thereby greatly reduce the complexity of code bases (cf. Brooks, 1986). Experience thus shows that the total complexity that can be represented with monolithic designs is much lower than with a modular design, and continuously adapting such models to new questions becomes harder and harder (Johanson & Hasselbring, 2018).

One example of an ecological model that was built with this technique is the GeMM model, which was designed to study eco-evolutionary dynamics of plant communities on islands (Leidinger et al., 2021). The original model design was well-suited to a number of questions relating to this study system (e.g. mechanisms of species invasions; Vedder, Leidinger and Sarmento Cabral, 2021). However, adapting the model to study terrestrial bird populations proved challenging and required the adaptation of large portions of code, making the whole code harder to understand (Vedder et al., 2022). Other examples of open-source models using the monolithic approach are the dispersal model of Sieger and Hovestadt (2021), the grazing models by Fust and Schlecht (2018) and Simon and Fortin (2020), or the butterfly model by Evans et al. (2019).

Degree of integration: Monolithic models offer maximal integration: every part of the code can access every other part.

Contributors: Monolithic models are typically built by individuals or small teams, who have a complete overview of the source code.

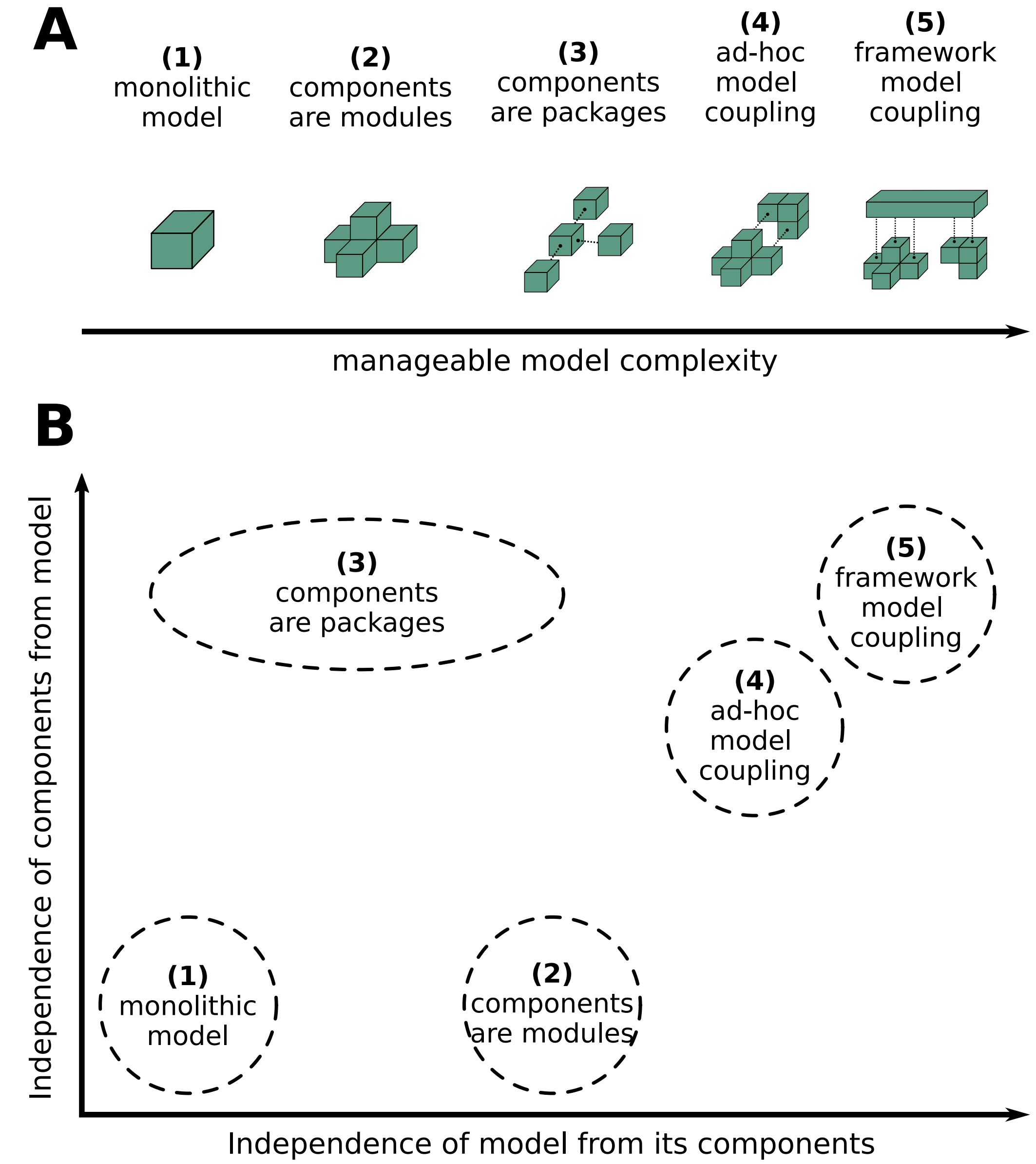

Figure 1: Different technical approaches to creating integrated models. The graph illustrates conceptually how independent components are of models and vice versa in each technique, i.e. how closely a specific component is tied to a specific model.

Can combine languages: Generally, monolithic models are built in a single programming language.

Components are model-independent: As monolithic models only have a single component, this component *is* the model and the two are not independent of each other in any way.

Difficulty of creating components: Creating a monolithic model is comparatively easy, as the software architecture is usually very simple and the overhead associated with packages or coupling frameworks is avoided. Also, although all parts have to be built by the developers, they have complete control over every aspect of their code.

Difficulty of coupling components: As monolithic models only have a single component, this question is not applicable.

Difficulty of extending model: Extending a monolithic model becomes progressively harder, as the lack of a clear internal structure leads to a rapidly mounting technical complexity that eventually precludes further development ("complexity ceiling", see glossary in Box 1).

3.2 Components are modules

Modellers that anticipate having to deal with large numbers of entities and processes may adopt a more modular approach (Bell et al., 2015). This is also important if the model is expected to grow beyond the scope of the original study. The simplest way of increasing modularity is by internally subdividing the code base and clearly specifying how the different modules interact with each other. This will generally include splitting up the code into multiple files and folders, and can be aided by programming language features such as Python or Julia modules or C/C++ header files, as well as by classes and interfaces in object-oriented programming languages such as Java.

This subdivision requires a little forethought, and developers must be careful to keep the interaction between modules as limited as possible (McConnell, 2004). Still, such an architecture is not hard to set up, and offers large flexibility for integration where that is necessary. Thus, this technique is well-suited to models that include multiple domains with multiple interactions. It probably works best for small to medium-sized models and developer teams, as the still comparatively high degree of integration likely will pose a significant complexity burden for very large models that are worked on by many different people or multiple teams.

One example of a model that applies this technique to good effect is ALMaSS, which simulates animal species in agricultural landscapes (Topping et al., 2003). It includes modules for multiple animal and crop species, a number that has been steadily growing over the last years (e.g. six animal species in Topping, 2011; seventeen in the 2022 code base, Topping, 2022). As another example, the macro-evolutionary model gen3sis uses a modular approach to allow the user to switch between different implementations of the simulated ecological and evolutionary processes (Hagen et al., 2021).

Degree of integration: Depending on the choice of programming language and modularisation technique (different files, classes, etc.), every part of the code can still access every other part where necessary. However, to keep up modularity, the developers should self-impose limits on the amount of integration they use depending on how much they consider necessary. This means that such models can have very different degrees of integration, but will generally be at a medium integration level.

Contributors: Modular models are typically built by small teams. Different team members will usually work on different sections, although all team members will have access to the complete source code.

Can combine languages: Generally, modular models are built in a single programming language.

Components are model-independent: Components in modular models are written specifically for the model they are a part of, and are therefore typically dependent on other parts of this particular model. Vice versa, the model does not have to be dependent on all of its components, with research questions often focussing on how enabling or disabling specific components (e.g. for land use dynamics) affects the overall model results.

Difficulty of creating components: Each component is intended to be small and self-contained, making the creation of components easy.

Difficulty of coupling components: Because the developers have full control over all parts of the source code, aligning components with each other is straightforward.

Difficulty of extending model: The greater degree of modularity makes adding new functionality or even new components much easier than with monolithic models. However, because of the still relatively high degree of integration, these models are likely to hit their complexity ceiling earlier than models with a stronger separation between components (see the following techniques).

3.3 Components are packages

All major programming languages support packages, also known as libraries. This is software that is designed to be included in other programs in order to perform a specific

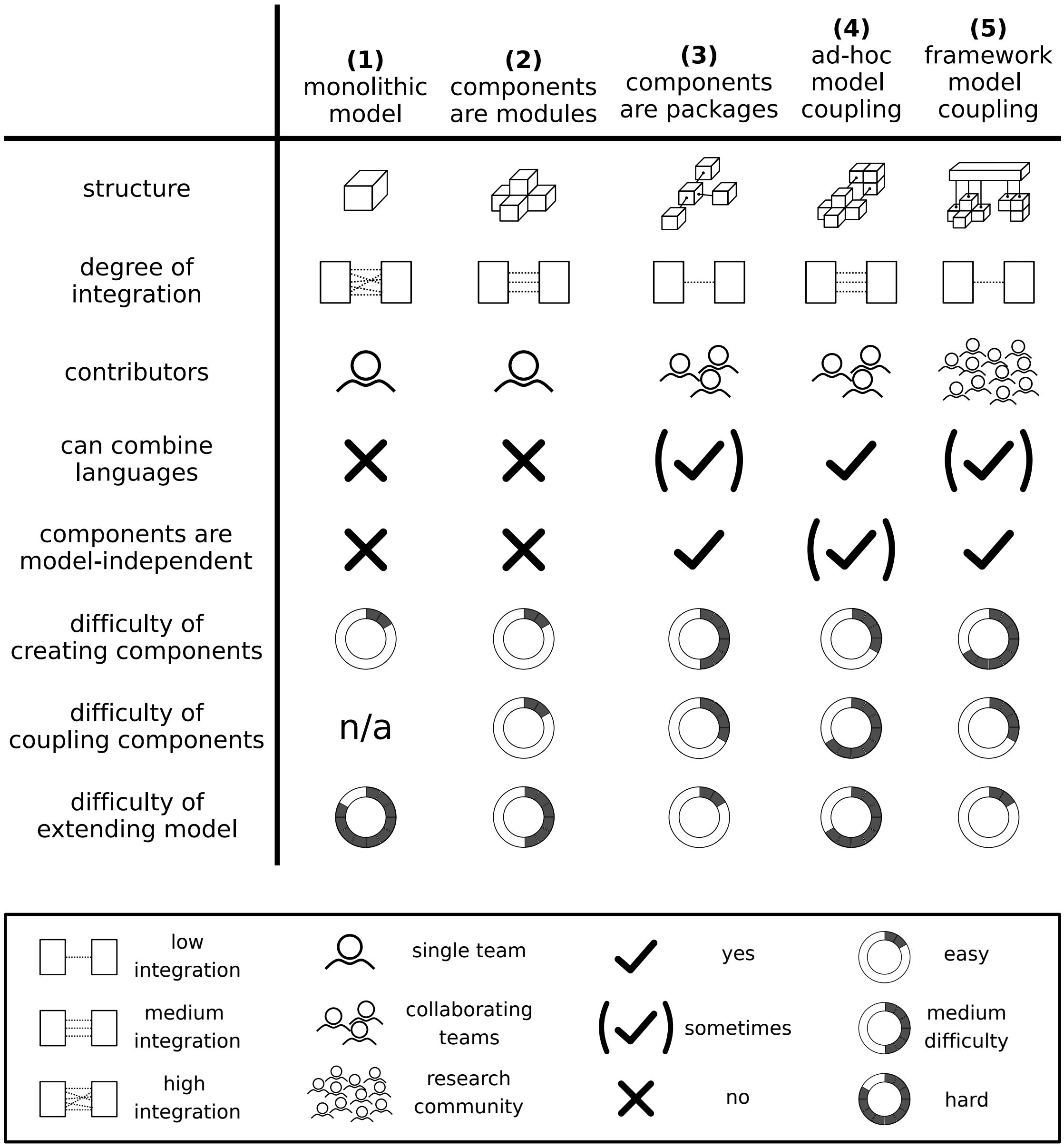

Figure 2: Different software engineering techniques have different advantages and drawbacks, making them suitable for different modelling purposes and contexts. This figure offers a qualitative comparison of a number of relevant points as a summary.

task, such as statistical analysis or visualisation. Packages are generally not built to be used as stand-alone executables. They are nonetheless independent of the programs they are embedded in, and are usable by any program with similar requirements. Thus, whereas modules (in the sense discussed above) are components that are created for and used solely within a specific model, packages are components that are meant to be used by many models. This is achieved by defining an API (application programmer interface) to specify the functions and classes that the given package provides, which input data it requires, and what output it produces. Because packages are largely self-contained, they provide a greater degree of modularity than modules. This becomes important as models grow in size, particularly if the model components are created by multiple developer teams.

By adhering to the specifications for creating packages in their chosen programming language, component developers can make their code interoperable and easily accessible to other model developers. Many programming languages offer package repositories to enable rapid dissemination of software, such as CRAN for R (cf. Ram et al., 2019), pip for Python (cf. Maji et al., 2020), and Pkg.jl for Julia (cf. Churavy et al., 2022). Using preexisting packages as model components allows model developers to greatly reduce the amount of work they have to do themselves, while automatically making the code more modular and thus more understandable. In some cases (particularly with compiled packages known as dynamically-linked libraries), it can also be possible to include code written in a different programming language in a model, thus allowing developers to benefit from a wider circle of previous work. On the downside, a package may not offer all the features the model developers would like, require the model code to be adjusted to fit the package requirements, or induce coding overhead for packaging and maintenance.

Although many ecologists are familiar with the concept of packages for data analysis (cf. Marwick et al., 2018), using packages as model components is not as widely spread. One example is the plant growth model by Schouten et al. (2020), who split up their model code into three independent packages that can also be used for other models. It is also an option to transform complete ecological models into packages, thus making them components that can be used by others. For example, this was done for the Madingley (Harfoot, Newbold et al., 2014; Hoeks et al., 2021) and RangeShifter models (Bocedi et al., 2021; Malchow et al., 2021) in order to make them interoperable with R.

Degree of integration: Packages are designed to offer a public interface, a set of functions and/or classes that can be called by users of the package, while making their internal workings inaccessible to users. This makes for strong modularity, but consequently low integration.

Contributors: Often, every package in a software project is developed by a separate team. Teams may or may not collaborate explicitly, e.g. by adjusting a package to provide functionality required for a specific application.

Can combine languages: Many programming languages can be combined with packages built in other languages, although this may be difficult to set up.

Components are model-independent: Packages are usually built to provide general functionality independent of a specific use case, and hence independent of the model they are included in. How dependent a model is on a given package depends on the availability of other packages offering equivalent functionality.

Difficulty of creating components: Every programming language has its own guidelines on how to create packages. Designing a good package API takes some thought, as it must consider both the current and future needs of package users as well as those of the developers.

Difficulty of coupling components: Loading packages into a software is designed to be simple, and packages often offer good documentation of their functionality. The largest difficulty comes from the model developers not having complete control over the source code of the package, so that they may have to adapt their model to the utilised packages.

Difficulty of extending model: Due to the high modularity of a package-based design, adding new functionality later on is not more difficult than the original model creation.

3.4 Ad-hoc model coupling

Sometimes, developers want to link two or more full models together to explore inter-domain effects and feedbacks. This is known as model coupling, and can be achieved by multiple technical means (Belete et al., 2017; Robinson et al., 2018). The simplest way is to adapt one or more of the models so that they can use each others' output files as input, and them run them sequentially if no feedbacks exist, or update them step-by-step in turn if feedbacks do exist. Instead of using files (which is simple, but likely to be very slow), the data exchange can also be implemented using a network connection, or by loading the components as packages (see above). In any case, the result is an integrated model whose components are themselves stand-alone models, but that have been adapted to specifically interact with each other. As this adaptation must be done for each set of models on a case-by-case basis, we refer to this form of coupling as "ad-hoc model coupling", to differentiate it from the more generic "framework model coupling" (see below).

Ad-hoc model coupling is well-suited for collaborations among a small number of development teams. Because the undertaken adaptation is specific to the target models, a relatively high degree of integration can be achieved. It is also often possible to couple models written in different programming languages. However, having to adapt two or more existing models to each other is finicky work that becomes rapidly more complex the more models are involved. Therefore, although this technique can be useful for coupling two or three models, it is not feasible for building large collections of interoperable models.

Ad-hoc model coupling was used, for example, by Synes et al. (2019) to combine the ecological movement model RangeShifter with the socio-economic land-use model CRAFTY. This allowed the authors to study the feedbacks between land use, crop yield, and pollinator abundance. Robinson et al. (2018) describe several other examples of ad-hoc model coupling, including a three-model coupling of the dynamic global vegetation model LPJ-GUESS with the climate model IMOGEN and the food system model PLUMv2. These were used to study interlinkages between agricultural intensification and expansion and climate change.

Degree of integration: While each component model is originally self-contained, the individualised coupling process allows developers to connect components quite closely to each other, thus achieving an intermediate degree of integration.

Contributors: Ad-hoc model coupling is generally used to bring together the work by a small number of developer teams.

Can combine languages: As ad-hoc coupling usually does not require source code compatibility (e.g. when coupling via input/output file exchange), models written in different programming languages can be combined as they are.

Components are model-independent: Because each component is itself a model, components are technically independent of each other. However, the coupling process may involve adapting the constituent models to be able to interact with each other, so there is a greater degree of mutual dependence than with framework coupling.

Difficulty of creating components: The difficulty of creating each component model depends strongly on its own complexity. Planning for model coupling already during the design phase of a model allows the component model to be kept minimal and at a lower level of complexity, thus reducing the difficulty of implementation.

Difficulty of coupling components: Coupling models on an ad-hoc basis can be tricky,

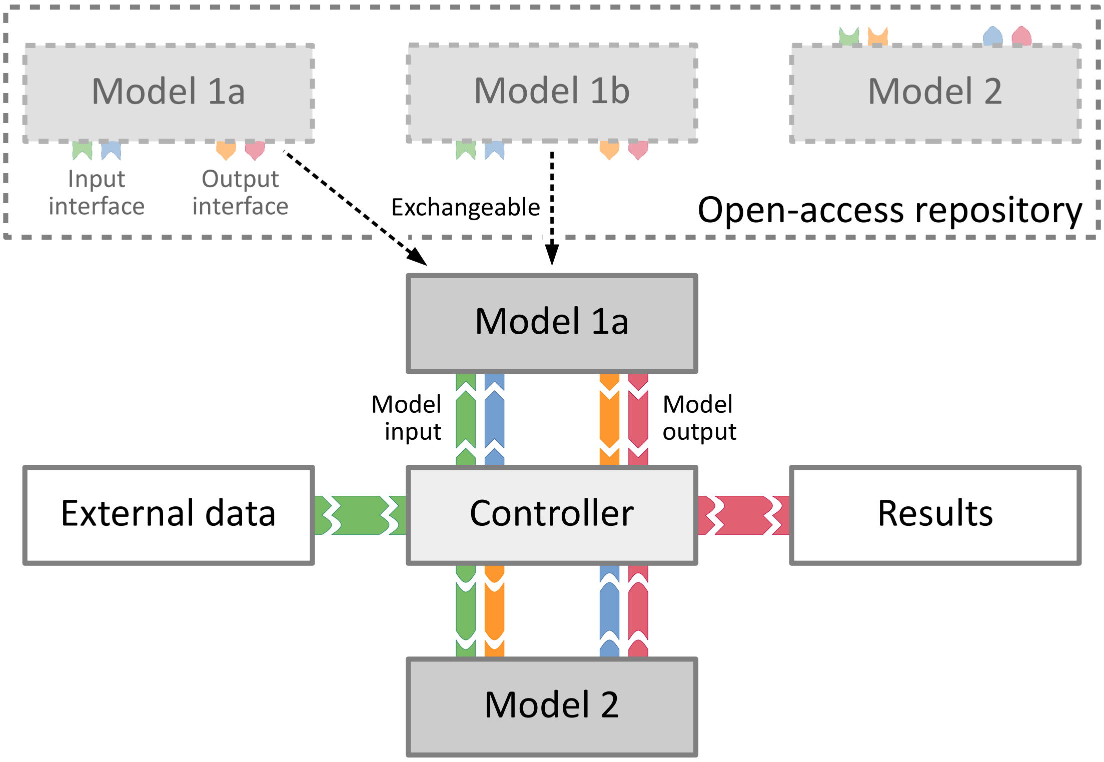

Figure 3: Conceptual depiction of a model coupling framework. Individual models are available from an open-access repository, and declare their required input and output variables using the standardised interface defined by the coupling framework. Modellers select relevant models from the repository (or build their own), and link them using a controller software provided by the coupling framework, which manages scheduling and data exchange.

as input and output data have to be carefully aligned. Modellers not only have to ensure that the component models are scientifically compatible (e.g. by converting between units or different representations of space), but also need to set up the technical communication between components (e.g. through file exchange or network connections).

Difficulty of extending model: Because the components are specifically aligned to each other, extending an ad-hoc coupled model further involves a lot of detail work to integrate new components with the existing structure. This may rapidly become unfeasible.

Box 2: Case study I: integrated models in climate/earth-system models

Climate models have been pushing the boundaries of what is computationally and scientifically possible since the 1950s (Edwards, 2011). Initially, these models were restricted to atmospheric processes, but soon began to be linked to other earth-system models, such as for the oceans or the cryosphere. In 2003, the Earth System Modelling Framework (ESMF) was introduced in order to encourage and facilitate this growing cooperation between institutes and disciplines (Hill et al., 2004). The ESMF consortium created a framework that could be used to integrate existing models into a single application. It worked by defining a "superstructure" (a basic software interface that component models could connect to) and providing an "infrastructure" (a collection of utility functions to help components communicate with each other). On the basis of this, the Earth System Prediction Suite (ESPS) was later set up to provide a curated collection of models that conform to the relevant standards and can therefore be expected to be readily interoperable (Theurich et al., 2016).

Importantly, the parallel development of multiple, competing modelling approaches fostered positive competitiveness and robustness in climate science, by exploring the range of possible future climate scenarios according to different models (J.-Y. Lee et al., 2021). This community approach to modelling has been formalised in the Coupled Model Intercomparison Project (CMIP; Eyring et al., 2016). The consistency with which greenhouse gas emissions were shown to drive climate change has given these modelling results a very high level of confidence with respect to the anthropogenic contribution to climate change. Thus, as simple as the question may be (e.g. "What is the effect of anthropogenic greenhouse gases?"), delivering a robust reply was very much achieved by integrating models, including complex interactions, and testing multiple models (Manabe, 2019). Moreover, only through such integration could tipping points be studied and further explored (Armstrong McKay et al., 2022).

This MIP approach (comparing the output of multiple models using standardised input data), pioneered by climate modelling, has since been extended to other modelling discipline. Examples include the Inter-Sectoral Impact Model Intercomparison Project (ISI-MIP, Rosenzweig et al., 2017), which also considers ecosystems, or the Biodiversity and Ecosystem Services Scenario-based Intercomparison of Models (BES-SIM, Kim et al., 2018).

3.5 Framework model coupling

The climate and earth science modelling communities were the first to realise the need for easy interoperability of diverse models, and pioneered the technique of framework model coupling (Box 1). A coupling framework defines a standard interface that all compatible models have to conform to, thus removing the need for case-by-case model adaptations. It also provides a central software to coordinate the execution of the coupled models (the controller), as well as utility functions to help with unit conversions, spatial and temporal scale alignment, and other practicalities of model coupling (Belete et al., 2017). This means that although model developers have to do some work to adapt their model to a given coupling framework, the model is then (at least in theory) compatible with any other model adapted to the same framework (Fig. 3). Achieving this kind of standardisation requires a very high level of coordination among researchers, but has proven highly valuable not just in climate and earth system modelling, but also in other modelling fields like agriculture (Box 2).

Multiple coupling frameworks are available, often originating in different modelling disciplines and differing in their specificity and ease of use (Knapen et al., 2013). The most general framework at the moment is probably OpenMI (Gregersen et al., 2007; Harpham et al., 2019), which was first developed by hydrological modellers but is now also used to link models in other fields, such as agriculture (S. Janssen et al., 2011), economics (Bulatewicz et al., 2010), or pathogen ecology (Shrestha et al., 2013). So far, however, coupling models using frameworks is still very rare in ecology, despite the technique's proven potential for achieving large-scale model integration as called for by Urban et al. (2022) and others.

Degree of integration: Adapting models to be used as components in framework coupling requires giving them a standardised API (similar to packages). Therefore, the interaction possibilities with each component are clearly defined and restricted, resulting in a strongly modular architecture with a medium to low degree of integration.

Contributors: The up-front cost of adapting a model to a framework is only worthwhile if many other models also use this framework. Therefore, establishing a coupling framework requires a high amount of coordination among many different teams of modellers. Once this barrier has been surmounted, the advantage is that any framework-compatible models can be easily linked, thus enabling collaboration across a wide range of teams and disciplines.

Can combine languages: Most frameworks support a specific set of programming languages. Models that want to use a framework therefore must either be written in one of these languages, or provide a wrapper for one of them (e.g. through an R or Python package).

Components are model-independent: In framework coupling, each component is a separate model that can also be used on its own, or in combination with other models in a different integrated model. Thus, there is maximum independence of the components from the model.

Difficulty of creating components: As each component is itself a model, the difficulty of creating it depends on its own type and complexity.

Difficulty of coupling components: Adjusting a component model to be compatible with a framework can be a bit of work, but once this has been done, coupling models within the framework is designed to be quick and easy.

Difficulty of extending model: Extending an integrated model is very simple, as long as any new components are compatible with the coupling framework.

4 Practical recommendations

Moving on from the overview of techniques, in this section we offer recommendations on how to use these techniques in practice. We are guided by three questions: (1) Which technique is best suited for my purposes? (2) How can I build components that are readily interoperable with components and models developed by others? (3) How can I build models that integrate existing components? We summarise our recommendations in table 1.

4.1 Choosing a technique

The first question to ask when selecting a technique at the beginning of a new modelling project is: How large and complex should the model become? The greater the desired complexity, the more important it is to use a more advanced technique (Fig. 1a; Johanson and Hasselbring, 2018). On the other hand, a more advanced technique may be unnecessarily complicated to use for a simple model. If a model is only intended to be used for a single, well-defined study, a monolithic model is likely to be the simplest and most efficient option (Section 3.1). If it should be used for multiple studies, and potentially further extended in future, a modular architecture is highly advisable. If all parts of the model can or should be written by the team itself, this would mean using

Box 3: Case study II: integrated models in agricultural sciences

Agriculture is another research field that has a long history of modelling. Several traditions of agricultural modelling may be identified. First, crop-growth models are used to predict yields under varying management regimes and environmental conditions (Pan & Chen, 2021). Secondly, farm models are used to assess (primarily economic) policy impacts both on individual farms and on the regional agricultural sector (Reidsma et al., 2018). Agricultural processes or systems are also included in models from fields such as human geography, environmental sciences, and ecology (e.g. Le et al., 2008; Schmidt et al., 2017; Topping et al., 2003).

Although coming from different disciplinary backgrounds, different model types are increasingly being combined in order to address questions that require multiple perspectives. For example, Piorr et al. (2009) coupled economic and environmental models to assess the likely effects of the European Common Agricultural Policy on production and ecosystem health, while Malawska and Topping (2018) did so to investigate the effects of market shifts on farmland species. More comprehensively, Schreinemachers and Berger (2011) created an integrated model with components for farm economics and technology, crop growth, water flow, and soil erosion and nutrients.

The merging of modelling disciplines is being encouraged by large-scale projects such as the EU's SEAMLESS, which aims to integrate socio-economic and environmental models across scales for policy assessments (S. Janssen et al., 2011; van Ittersum et al., 2008), or the Agricultural Model Intercomparison and Improvement Project (AgMIP), which seeks to bring together crop-growth, farm, and climate models (Rosenzweig et al., 2013). To help this exchange among modellers, the US Department of Agriculture pushed development of the Object Modelling System (OMS), a lightweight environmental modelling framework similar to the ESMF (David et al., 2013), while the Agricultural Model Exchange Initiative (AMEI) aims to establish a modelling community similar to the ESPS (Enders et al., 2018). Although the use of scientific models in agricultural policy-making is still limited, these efforts are establishing a base of knowledge that is increasingly being drawn on for policy impact assessments by governments and international organisations and agreements, for instance in the European Union (Reidsma et al., 2018).

Table 1: Practical recommendations for creating large, integrated models, or components for such models. See section 4 for details.

Aim	**Recommendations**
choosing a technique	▪ identify how complex the model should be ▪ identify whether suitable components are already available ▪ examine what software expertise is / is not available in the team of collaborators ▪ envision how the model should or could be used in future
building interoperable components	▪ follow good coding practice ▪ provide detailed documentation ▪ make source code public ▪ provide software as a package ▪ provide bindings to coupling frameworks
building integrated models	▪ choose modular software architectures ▪ build on existing software where possible ▪ collaborate with other modellers ▪ learn from other modelling disciplines ▪ utilise coupling frameworks

"components as modules" (Section 3.2); if existing libraries or packages can be used it becomes "components as packages" (Section 3.3). For larger models that integrate across domains, coupling with existing models can be helpful or even essential, either with ad-hoc coupling (Section 3.4) or with framework coupling (Section 3.5).

For any medium- or larger-sized model, the second question is therefore to find out whether there are suitable components or models already available that can be utilised (either as packages, or for model coupling). These can be searched for using e.g. Google Scholar, Github, or CoMSES (Rollins et al., 2014). To assess whether a component or model is suitable, the following questions can help: Does it include the processes and state variables that are relevant to my research question? Do I have the required input data available? Is the software open source, and is it well-documented? Does it have an active user community, and/or are the developers easy to contact? When deciding whether or not to use externally developed software components, one must weigh the cost of working with software that may not be a perfect match for the project requirements against the cost of having to develop the desired functionality internally ("buy vs. build", cf. Brooks, 1986).

The third question is the available expertise and experience in software development among the team members. While monolithic models (Section 3.1) can be constructed by anyone who knows how to program, the other techniques require a little more knowledge of software engineering principles and tools. In particular framework model coupling (Section 3.5) has quite a steep learning curve. If the study question requires one of the more advanced techniques, researchers may decide to invest the time to learn the necessary skills themselves, collaborate with other researchers who have the expertise, or employ professional developers (Cohen et al., 2021).

Finally, it is good to have at least a vague idea of how the model to be built could or should be used in future. The longer it is to be used and the more people are to use it, the more important it is to use one or several of the more advanced techniques (cf. the experiences with the FORMIND model, Box 4). Although these come with a greater up-front cost in designing the software architecture and setting up the code base, they make it significantly easier to extend the model in future and to use it together with software written by other researchers (Fig. 2).

4.2 Building interoperable components

An important insight is that the road to large complex models starts with the development of small interoperable ones. Therefore, if as a modelling community we want to work towards developing larger models, individual modellers need to become better at building components and models that can easily be linked up with others (Bell et al., 2015; Berger et al., 2024).

To build components that can be readily integrated into large models, the aim should be to create high-quality scientific software products that are as easy as possible for other researchers to (re-)use (McIntire et al., 2022; Sanders & Kelly, 2008). This requires rigorous application of the FAIR criteria: Findable, Accessible, Interoperable, and Reusable (Barton et al., 2022; Hasselbring et al., 2020). Essentially, this boils down to learning and following best practices for software development in computational science in general (Balaban et al., 2021; Wilson et al., 2014) and ecological modelling in particular (Ropella et al., 2002; Scheller et al., 2010; Vedder, Ankenbrand & Cabral, 2021).

Important principles here are writing clean code (Filazzola & Lortie, 2022), using version control for open-source development (Perez-Riverol et al., 2016), using automated testing and code reviews for verification (Holzworth et al., 2011; Vable et al., 2021), and providing good technical and scientific documentation (Grimm, Railsback et al., 2020; B. D. Lee, 2018). Communities such as rOpenSci can help modellers write better software by providing detailed technical reviews (Ram et al., 2019). Creating high-quality software is important to make the code accessible and usable by others. It also makes code easier to extend by avoiding "technical debt", i.e. unnecessary technical complexity that slows down future development (see Glossary).

It is crucial that ecological modellers make all their source code openly available and document it sufficiently, not only for the sake of scientific reproducibility but also to enable others to build on the components they created (M. A. Janssen et al., 2020). This can be done on general-purpose platforms like Github (for collaborative development) or Zenodo (for code archiving). More narrowly modelling-oriented platforms can provide additional benefits, by acting as central repositories that make finding useful models and components quick and easy (Bell et al., 2015). One such platform is CoMSES, a library of agent-based and individual-based models that is seeing widespread use in the social sciences but is still little known among ecologists (M. A. Janssen et al., 2008; Rollins et al., 2014).

Modellers should also form the habit of making their software available as installable packages in their preferred programming language, ideally in a standard package repository (e.g. PyPI for Python or CRAN for R). This lowers the bar for installing components considerably, making it easier for other researchers to build on existing work.

To improve dissemination of new research software, several journals now allow developers to publish descriptions of applications and packages as a separate article type (e.g. *Methods in Ecology and Evolution*, *Journal of Open Source Software*). This has the additional benefit of providing scientific incentive (i.e. publications) for releasing code, which is particularly valuable for early-career researchers.

As ecological models become larger and more complex, the importance of coupling frameworks will increase. This is particularly true as ecologists seek to couple their models to models from fields that already use such frameworks, including the physical and social sciences. Therefore, ecologists should learn about existing frameworks and acquaint themselves with their use (Belete et al., 2017; Knapen et al., 2013). Seen from the perspective of a component developer, making a component compatible with an established coupling framework would be a major step toward interdisciplinary model interoperability.

Finally, model interoperability would be greatly aided by the adoption of a unified set of Essential Biodiversity Variables as standardised input/output variables for models (Jetz et al., 2019). This would enable harmonisation of model results and ease model coupling, as well as make it easier to interface models with empirical data sources (Fer et al., 2021; Urban et al., 2022). A coordinated move in this direction would be especially effective if accompanied by a standardisation of metadata, such as for the recently proposed Reusable Building Blocks (Berger et al., 2024).

4.3 Building integrated models

As ecologists think about creating large, complex models integrating many different entities, processes, and domains (e.g. Urban et al., 2016; Urban et al., 2022), they need to think very carefully about the technical challenges this entails. Building large software well is a significant undertaking, especially as software complexity does not scale linearly with software size (McConnell, 2004). Research teams need to consider the resources that are available for development, including time, money, manpower, and know-how. Based on this, they must decide what level of scientific complexity can be feasibly modelled, and which integration technique is most suitable to their purpose and context (Section 3, 4.1).

When attempting to build large models, researchers must realise the importance of carefully planning the code's design and architecture. For this, it is useful to have team members who have trained in software engineering (Cohen et al., 2021). Studying well-known open-source programs is also an excellent way to learn more about good software architecture and current practices in software development (e.g. Brown & Wilson, 2011).

The experience of other modelling fields shows that eventually, model complexity grows to such a degree that coupling frameworks are often the most effective way to further growth (Box 2,3). Ecological modelling does not seem to have reached this point yet, but considering the increasing amount of integration of ecological models with climate, hydrological, economic, sociological and other models, it can be expected that the field will reach this stage soon. This is a promising development, but one that brings its own set of challenges. Model coupling not only involves adaptation of existing software, but also requires models to be made scientifically compatible with regards to spatial and temporal scales and input and output data (Belete et al., 2017; Brandmeyer & Karimi, 2000). Coupling frameworks can greatly help with this process, but add their own complexity overhead. Complicating matters is the range of available coupling frameworks, each established in different disciplines and available for different programming languages.

It should be noted that it is not an aim that all ecological models should be large and integrated, or that all modellers need to work with framework model coupling. Smaller models have their place (Sun et al., 2016), and all techniques can be useful (Section 4.1). However, as argued above, in addition to the small and medium-sized models we already have, we will need larger and integrated models to study the interdisciplinary questions now facing science.

In view of all this, it seems an important challenge for the next years of ecological modelling to become better at building these integrated models. This will include training modellers in software development, as well as establishing conventions and standards for model publication and coupling. While other fields have been successfully using coupled models for decades, such coupling is rarely done in ecological modelling. This is an area where the recently-formed Open Modelling Foundation could be a vital catalyst for future methodological developments (Barton et al., 2022).

5 Discussion

5.1 The need for large integrated models

Computational models have become essential instruments for scientific research and policy advice. They play a central role in the work of both the IPCC (e.g. Eyring et al., 2016) and IPBES (e.g. IPBES, 2016), helping us better understand the global crises of climate change and biodiversity loss, and the interlinkages between them (Pörtner et al., 2021). However, the use and acceptance of models is much better established in climate change policy than in biodiversity policy (Urban et al., 2022). This is partly due to insufficient and misaligned communication between modellers and decision-makers (IPBES, 2016; Will et al., 2021), but also because biodiversity is often not well integrated into models of social-ecological systems (Vedder et al., submitted).

In the face of an ongoing ecological crisis, ecologists seek not only to understand Global Change processes but also to provide relevant and timely advice to policy makers and stakeholders. To do so, we will need increasingly advanced models that can provide insight into the multi-scale, multi-domain systems that we study. This requires expanding our models beyond their current focus on a small number of ecological processes, and embrace the complexity of real-world systems with their multi-scale pressures (Topping et al., 2015; Urban et al., 2022). This can only be effectively done by developing a practice of interdisciplinary modelling.

This is relevant in two directions. First, ecological modellers need to link their models to models of other physical systems. There is a strong tradition of mechanistic modelling in several disciplines in the environmental sciences, most notably in climate science, but also in areas like hydrology, ecotoxicology, and earth system science more generally. Ecologists can profit from and contribute to existing model coupling initiatives, such as OpenMI (Harpham et al., 2019), various Integrated Assessment Models (Harfoot, Tittensor et al.,

Box 4: Case study III: life history of a large ecological model

FORMIND (Bohn et al., 2014; Fischer et al., 2016; Köhler & Huth, 1998) is a process-based forest model incorporating regeneration, competition, growth, and mortality of trees. As individuals are modelled explicitly, the model allows a detailed analysis of forest structure and productivity under varying environmental conditions. FORMIND has a modular design, allowing the base model to be adjusted to a wide range of research questions, including the effects of forests fragmentation (Pütz et al., 2011), landslides (Dislich & Huth, 2012), water competition and droughts (Gutiérrez et al., 2014), and wild fires (Fischer, 2021).

The components are coupled via the "**components are modules**" strategy. Nonetheless, model and components remain strongly linked, to facilitate the exchange of data and the integration of components into the model's main routines. This architecture made it easy to add new components to the model without major changes to the historic core. However, with a growing number of modules, the code base became progressively more complex, making it increasingly difficult to update the model without breaking functionality.

In recent years, FORMIND has been combined with external software via **ad-hoc coupling** methods, e.g. to fit the model to field data (Lehmann & Huth, 2015). Building on the model's text file interface for data exchange reduced the need for code changes but also limited flexibility and created a significant computational overhead. Hence, a Python package wrapping the original C++ code is under development, allowing users to read and manipulate parameters and state variables at runtime ("**components are packages**"; first application in [Fischer et. al., manuscript in preparation]).

Currently, FORMIND is also being coupled with an external soil moisture model (Samaniego et al., 2010) to study forest-soil interactions. As the two models run at different temporal and spatial scales and efficient data exchange is required for the intended large-scale simulations, the Python-based framework FINAM (Lange et al., 2023) is used to couple the models ("**framework model coupling**"). Implementing the FINAM interface for FORMIND did not require changes to the existing model and only minimal additional code. As the FINAM interface is model-agnostic, the interface can also facilitate the integration of FORMIND into comprehensive landscape-scale models of the environment (cf. Cabral et al., 2023; Urban et al., 2022).

2014), or ISI-MIP (Rosenzweig et al., 2017).

Secondly, ecological modellers should become more involved in the burgeoning field of socio-ecological systems research. There have been significant recent advances in coupling ecological and socio-economic models (e.g. Guillem et al., 2015; Synes et al., 2019). Still, there are many challenges in incorporating mutual feedbacks between human and natural systems (Farahbakhsh et al., 2022), and in representing the multiple scales relevant to many socio-ecological systems (Lippe et al., 2019). Altogether, scaling up our understanding of socio-ecological systems, including telecoupled systems, cannot be achieved without much more complex models than currently exist.

Ultimately, addressing sustainability challenges requires models that combine all three domains: the ecological, the physical, and the human. For instance, when studying land use change, it is desirable to combine models from multiple fields, including human geography, ecology, and climate science, in order to understand how processes in each of the three domains influence processes in the other two (Cabral et al., 2023).

5.2 A way forward

To meet the scientific and technical challenges of creating large integrated models, ecological modellers should invest into software engineering training, as well as into intra- and interdisciplinary standardisation processes.

In this paper we have introduced a set of software engineering techniques for building models of different sizes and complexities. Which technique is best suited to a given research context depends on the aim of the model, the available development resources, and the likely future users of the model. The larger a model becomes, the more important it is to carefully plan its software architecture and to follow good software engineering practices (Johanson & Hasselbring, 2018). As a model grows over time, it is also likely that new techniques will become relevant, as exemplified by the forest model FORMIND (Box 3).

Like many other computational scientists, ecological modellers face the problem that few have had good training in software engineering (Nowogrodzki, 2019). Although this is understandable given the contents of most ecology curricula in universities (Farrell & Carey, 2018), it frequently results in code that ignores the most basic principles of software quality and validation (Prabhu et al., 2011). This issue not only undermines the credibility of our scientific models, but also impedes researchers' ability to utilize and expand upon existing software. (Sanders & Kelly, 2008). It is therefore imperative that we invest into better software training for modellers, and collaborate with professional software developers to produce reliable and usable model code (Cohen et al., 2021).

We recognise that this currently faces a number of institutional barriers. Some relate to finances: smaller research groups often lack the funds to hire professional developers, and getting funding to develop and maintain research software can be difficult (Nowogrodzki, 2019). Furthermore, research institutes often cannot offer the job security or salaries that are competitive with those of software developers in industry. Other barriers relate to culture: because research software is often not valued in itself, but only for the scientific output it produces, little importance is attached to tasks that make the software itself better, such as improving code quality and ensuring appropriate documentation (Johanson & Hasselbring, 2018). Especially early-career researchers in modelling may be pressured to "produce results" as fast as possible, leaving them little time to gain the technical skills needed to produce more complex software. This in turn contributes to the "yet another model" syndrome, i.e. the observed profusion of simple models at the expense of more advanced ones (O'Sullivan et al., 2016). We therefore propose that to create more complex models, ecological modellers should invest in becoming better software developers. This will require a cultural shift to value code as a scientific output in its own right (Mislan et al., 2016).

Beyond simply improving the quality of model software, ecological modellers also need to think more about standardisation of data and metadata, both within ecology and with other disciplines. Within ecology, increased use of online model repositories such as CoMSES would aid the discoverability of models and model components (Bell et al., 2015; Rollins et al., 2014). Further development and use of standardised Essential Biodiversity Variables could help to align the input and output variables of different models, and of models with empirical data sources, thus allowing easier coupling (Urban et al., 2022). Programs such as BES-SIM can help harmonise the results of ecological models, and contribute to a more unified and strategic development of the field (Kim et al., 2018). To collaborate with disciplines outside ecology, ecological modellers need to learn more about existing standards, frameworks, and collaboration networks. Here, the Open Modelling Foundation is a valuable initiative to bring together modellers from numerous domains to promote collaboration and standardisation (Barton et al., 2022). Not least, we can use such connections to learn about how other modelling disciplines have solved the challenges our field currently faces, as many of these disciplines are methodologically much further advanced than ecological modelling.

6 Conclusion

To advance ecological modelling, we will need to create larger and more integrated mechanistic models. This is particularly urgent in light of the growing demand for interdisciplinary modelling to face the dual crises of climate change and biodiversity loss. We must learn not only to build comprehensive models of biodiversity and ecosystems, but also to couple these with climate, land use, economic, and other models.

In this review, we highlight the inherent trade-off between integration and modularity, and explain the associated tension between scientific requirements and technical constraints. We showcase five different integration techniques and how they are currently being used, as well as discussing their relative strengths and weaknesses. As practical recommendations, we emphasise that ecological modellers need more training in software engineering, must adopt FAIR research practices, and should begin to think about how best to apply existing coupling techniques to ecological models.

Acknowledgements

The authors wish to thank Quillon Harpham and three anonymous reviewers for their valuable inputs on the topic. DV is funded through the project CAP4GI by the Federal Ministry of Education and Research (BMBF), within the framework of the Strategy „Research for Sustainability" (FONA, www.fona.de/en) as part of its Social-Ecological Research funding priority, funding no. 01UT2102A. Responsibility for the content of this publication lies with the authors. DV and GP gratefully acknowledge the support of iDiv, funded by the German Research Foundation (DFG–FZT 118, 202548816). SMF gratefully acknowledges funding received from the German Research Foundation (DFG; project ForcTrait).

Author contributions

DV and SMF conceived the idea for the review and developed the figures; DV led the writing of the manuscript; all authors contributed critically to the drafts and gave final approval for publication.

References

Abelson, H., Sussman, G. J., & Sussman, J. (1996). *Structure and interpretation of computer programs* (2. ed). MIT Press.
OCLC: 34576857.

Armstrong McKay, D. I., Staal, A., Abrams, J. F., Winkelmann, R., Sakschewski, B., Loriani, S., Fetzer, I., Cornell, S. E., Rockström, J., & Lenton, T. M. (2022). Exceeding 1.5°C global warming could trigger multiple climate tipping points. *Science*, *377*(6611), eabn7950. https://doi.org/10.1126/science.abn7950

Balaban, G., Grytten, I., Rand, K. D., Scheffer, L., & Sandve, G. K. (2021). Ten simple rules for quick and dirty scientific programming. *PLOS Computational Biology*, *17*(3), e1008549. https://doi.org/10.1371/journal.pcbi.1008549

Barton, C. M., Lee, A., Janssen, M. A., Porter, C., Greenberg, J., Swantek, L., Frank, K., Chen, M., & Jagers, H. R. A. (2022). How to make models more useful. *Proceedings of the National Academy of Sciences*, *119*(35), 4.

Beck, F., & Diehl, S. (2011). On the congruence of modularity and code coupling. *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, 354–364. https://doi.org/10.1145/2025113.2025162

Belete, G. F., Voinov, A., & Laniak, G. F. (2017). An overview of the model integration process: From pre-integration assessment to testing. *Environmental Modelling & Software*, *87*, 49–63. https://doi.org/10.1016/j.envsoft.2016.10.013

Bell, A. R., Robinson, D. T., Malik, A., & Dewal, S. (2015). Modular ABM development for improved dissemination and training. *Environmental Modelling & Software*, *73*, 189–200. https://doi.org/10.1016/j.envsoft.2015.07.016

Berger, U., Bell, A., Barton, C. M., Chappin, E., Dreßler, G., Filatova, T., Fronville, T., Lee, A., van Loon, E., Lorscheid, I., Meyer, M., Müller, B., Piou, C., Radchuk, V., Roxburgh, N., Schüler, L., Troost, C., Wijermans, N., Williams, T. G., . . . Grimm, V. (2024). Towards reusable building blocks for agent-based modelling and theory development. *Environmental Modelling & Software*, *175*, 106003. https://doi.org/10.1016/j.envsoft.2024.106003

Bocedi, G., Palmer, S. C. F., Malchow, A.-K., Zurell, D., Watts, K., & Travis, J. M. J. (2021). RangeShifter 2.0: An extended and enhanced platform for modelling spatial eco-evolutionary dynamics and species' responses to environmental changes. *Ecography*, *44*(10), 1453–1462. https://doi.org/10.1111/ecog.05687

Bohn, F. J., Frank, K., & Huth, A. (2014). Of climate and its resulting tree growth: Simulating the productivity of temperate forests. *Ecological Modelling*, *278*, 9–17. https://doi.org/10.1016/j.ecolmodel.2014.01.021

Brandmeyer, J. E., & Karimi, H. A. (2000). Coupling methodologies for environmental models. *Environmental Modelling & Software*, *15*(5), 479–488. https://doi.org/10.1016/S1364-8152(00)00027-X

Brooks, F. (1986). No Silver Bullet – Essence and Accident in Software Engineering. In H.-J. Kugler (Ed.), *Proceedings of the IFIP Tenth World Computing Conference* (pp. 1069–1076). Elsevier Science B.V.

Brown, A., & Wilson, G. (Eds.). (2011). *The Architecture of Open Source Applications.* Creative Commons.
OCLC: 936178393.

Bulatewicz, T., Yang, X., Peterson, J. M., Staggenborg, S., Welch, S. M., & Steward, D. R. (2010). Accessible integration of agriculture, groundwater, and economic models using the Open Modeling Interface (OpenMI): Methodology and initial results. *Hydrol. Earth Syst. Sci.*, 14.

Cabral, J. S., Mendoza-Ponce, A., da Silva, A. P., Oberpriller, J., Mimet, A., Kieslinger, J., Berger, T., Blechschmidt, J., Brönner, M., Classen, A., Fallert, S., Hartig, F., Hof, C., Hoffmann, M., Knoke, T., Krause, A., Lewerentz, A., Pohle, P., Raeder, U., … Zurell, D. (2023). The road to integrate climate change projections with regional land-use–biodiversity models. *People and Nature*, *n/a*(n/a). https://doi.org/10.1002/pan3.10472

Cabral, J. S., Valente, L., & Hartig, F. (2017). Mechanistic simulation models in macroecology and biogeography: State-of-art and prospects. *Ecography*, *40*(2), 267–280. https://doi.org/10.1111/ecog.02480

Churavy, V., Godoy, W. F., Bauer, C., Ranocha, H., Schlottke-Lakemper, M., Räss, L., Blaschke, J., Giordano, M., Schnetter, E., Omlin, S., Vetter, J. S., & Edelman, A. (2022, November 10). *Bridging HPC Communities through the Julia Programming Language.* arXiv: 2211.02740 [cs]. https://doi.org/10.48550/arXiv.2211.02740

Cohen, J., Katz, D. S., Barker, M., Hong, N. C., Haines, R., & Jay, C. (2021). The Four Pillars of Research Software Engineering. *IEEE Software*, *38*(1), 97–105. https://doi.org/10.1109/MS.2020.2973362

David, O., Ascough, J. C., Lloyd, W., Green, T. R., Rojas, K. W., Leavesley, G. H., & Ahuja, L. R. (2013). A software engineering perspective on environmental modeling framework design: The Object Modeling System. *Environmental Modelling & Software*, *39*, 201–213. https://doi.org/10.1016/j.envsoft.2012.03.006

DeAngelis, D. L., & Grimm, V. (2014). Individual-based models in ecology after four decades. *F1000prime reports*, *6*, 39. https://doi.org/10.12703/P6-39

Dijkstra, E. W. (1972). The humble programmer. *Communications of the ACM*, *15*(10), 859–866. https://doi.org/10.1145/355604.361591

Dislich, C., & Huth, A. (2012). Modelling the impact of shallow landslides on forest structure in tropical montane forests. *Ecological Modelling*, *239*, 40–53. https://doi.org/10.1016/j.ecolmodel.2012.04.016

Edwards, P. N. (2011). History of climate modeling. *WIREs Climate Change*, *2*(1), 128–139. https://doi.org/10.1002/wcc.95

Enders, A., Martre, P., Raynal, H., Athanasiadis, I., Donatelli, M., Fumagalli, D., Holzworth, D., Stöckle, C., & Hoogenboom, G. (2018). Agricultural Model Exchange Initiative (AMEI), 61. https://scholarsarchive.byu.edu/iemssconference/2018/Stream-A/61

Evans, L. C., Sibly, R. M., Thorbek, P., Sims, I., Oliver, T. H., & Walters, R. J. (2019). Quantifying the effectiveness of agri-environment schemes for a grassland butterfly using individual-based models. *Ecological Modelling*, *411*, 108798. https://doi.org/10.1016/j.ecolmodel.2019.108798

Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., & Taylor, K. E. (2016). Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization. *Geoscientific Model Development*, *9*(5), 1937–1958. https://doi.org/10.5194/gmd-9-1937-2016

Farahbakhsh, I., Bauch, C. T., & Anand, M. (2022). Modelling coupled human–environment complexity for the future of the biosphere: Strengths, gaps and promising directions. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *377*(1857), 20210382. https://doi.org/10.1098/rstb.2021.0382

Farrell, K. J., & Carey, C. C. (2018). Power, pitfalls, and potential for integrating computational literacy into undergraduate ecology courses. *Ecology and Evolution*, *8*(16), 7744–7751. https://doi.org/10.1002/ece3.4363

Fer, I., Gardella, A. K., Shiklomanov, A. N., Campbell, E. E., Cowdery, E. M., De Kauwe, M. G., Desai, A., Duveneck, M. J., Fisher, J. B., Haynes, K. D., Hoffman, F. M., Johnston, M. R., Kooper, R., LeBauer, D. S., Mantooth, J., Parton, W. J., Poulter, B., Quaife, T., Raiho, A., … Dietze, M. C. (2021). Beyond ecosystem modeling: A roadmap to community cyberinfrastructure for ecological data-model integration. *Global Change Biology*, *27*(1), 13–26. https://doi.org/10.1111/gcb.15409

Filazzola, A., & Lortie, C. (2022). A call for clean code to effectively communicate science. *Methods in Ecology and Evolution*, *n/a*(n/a). https://doi.org/10.1111/2041-210X.13961

Fischer, R. (2021). The Long-Term Consequences of Forest Fires on the Carbon Fluxes of a Tropical Forest in Africa. *Applied Sciences*, *11*(10), 4696. https://doi.org/10.3390/app11104696

Fischer, R., Bohn, F., Dantas De Paula, M., Dislich, C., Groeneveld, J., Gutiérrez, A. G., Kazmierczak, M., Knapp, N., Lehmann, S., Paulick, S., Pütz, S., Rödig, E., Taubert, F., Köhler, P., & Huth, A. (2016). Lessons learned from applying a forest gap model to understand ecosystem and carbon dynamics of complex tropical forests. *Ecological Modelling*, *326*, 124–133. https://doi.org/10.1016/j.ecolmodel.2015.11.018

Fust, P., & Schlecht, E. (2018). Integrating spatio-temporal variation in resource availability and herbivore movements into rangeland management: RaMDry—An agent-based model on livestock feeding ecology in a dynamic, heterogeneous, semi-arid environment. *Ecological Modelling*, *369*, 13–41. https://doi.org/10.1016/j.ecolmodel.2017.10.017

Gregersen, J. B., Gijsbers, P. J. A., & Westen, S. J. P. (2007). OpenMI: Open modelling interface. *Journal of Hydroinformatics*, *9*(3), 175–191. https://doi.org/10.2166/hydro.2007.023

Grimm, V., Ayllón, D., & Railsback, S. F. (2017). Next-Generation Individual-Based Models Integrate Biodiversity and Ecosystems: Yes We Can, and Yes We Must. *Ecosystems*, *20*(2), 229–236. https://doi.org/10.1007/s10021-016-0071-2

Grimm, V., Johnston, A. S. A., Thulke, H.-H., Forbes, V. E., & Thorbek, P. (2020). Three questions to ask before using model outputs for decision support. *Nature Communications*, *11*(1), 4959. https://doi.org/10.1038/s41467-020-17785-2

Grimm, V., & Railsback, S. F. (2005). *Individual-based Modeling and Ecology*. Princeton University Press.

Grimm, V., Railsback, S. F., Vincenot, C. E., Berger, U., Gallagher, C., DeAngelis, D. L., Edmonds, B., Ge, J., Giske, J., Groeneveld, J., Johnston, A. S. A., Milles, A., Nabe-Nielsen, J., Polhill, J. G., Radchuk, V., Rohwäder, M.-S., Stillman, R. A., Thiele, J. C., & Ayllón, D. (2020). The ODD Protocol for Describing Agent-Based and Other Simulation Models: A Second Update to Improve Clarity, Replication, and Structural Realism. *Journal of Artificial Societies and Social Simulation*, *23*(2), 7. https://doi.org/10.18564/jasss.4259

Guillem, E., Murray-Rust, D., Robinson, D., Barnes, A., & Rounsevell, M. (2015). Modelling farmer decision-making to anticipate tradeoffs between provisioning ecosystem services and biodiversity. *Agricultural Systems*, *137*, 12–23. https://doi.org/10.1016/j.agsy.2015.03.006

Gutiérrez, A. G., Armesto, J. J., Díaz, M. F., & Huth, A. (2014). Increased Drought Impacts on Temperate Rainforests from Southern South America: Results of a Process-Based, Dynamic Forest Model (B. Hérault, Ed.). *PLoS ONE*, *9*(7), e103226. https://doi.org/10.1371/journal.pone.0103226

Hagen, O., Flück, B., Fopp, F., Cabral, J. S., Hartig, F., Pontarp, M., Rangel, T. F., & Pellissier, L. (2021). Gen3sis: A general engine for eco-evolutionary simulations of the processes that shape Earth's biodiversity. *PLOS Biology*, *19*(7), e3001340. https://doi.org/10.1371/journal.pbio.3001340

Harfoot, M. B. J., Newbold, T., Tittensor, D. P., Emmott, S., Hutton, J., Lyutsarev, V., Smith, M. J., Scharlemann, J. P. W., & Purves, D. W. (2014). Emergent Global Patterns of Ecosystem Structure and Function from a Mechanistic General Ecosystem Model. *PLOS Biology*, *12*(4), e1001841. https://doi.org/10.1371/journal.pbio.1001841

Harfoot, M. B. J., Tittensor, D. P., Newbold, T., McInerny, G., Smith, M. J., & Scharlemann, J. P. W. (2014). Integrated assessment models for ecologists: The present and the future. *Global Ecology and Biogeography*, *23*(2), 124–143. https://doi.org/10.1111/geb.12100

Harpham, Q., Hughes, A., & Moore, R. (2019). Introductory overview: The OpenMI 2.0 standard for integrating numerical models. *Environmental Modelling & Software*, *122*, 104549. https://doi.org/10.1016/j.envsoft.2019.104549

Hasselbring, W., Carr, L., Hettrick, S., Packer, H., & Tiropanis, T. (2020). From FAIR research data toward FAIR and open research software. *it - Information Technology*, *62*(1), 39–47. https://doi.org/10.1515/itit-2019-0040

Hill, C., DeLuca, C., Balaji, Suarez, M., & Da Silva, A. (2004). The architecture of the earth system modeling framework. *Computing in Science & Engineering*, *6*(1), 18–28. https://doi.org/10.1109/MCISE.2004.1255817

Hoeks, S., Tucker, M. A., Huijbregts, M. A. J., Harfoot, M. B. J., Bithell, M., & Santini, L. (2021). MadingleyR: An R package for mechanistic ecosystem modelling. *Global Ecology and Biogeography*, *30*(9), 1922–1933. https://doi.org/10.1111/geb.13354

Holzworth, D. P., Huth, N. I., & deVoil, P. G. (2011). Simple software processes and tests improve the reliability and usefulness of a model. *Environmental Modelling & Software*, *26*(4), 510–516. https://doi.org/10.1016/j.envsoft.2010.10.014

IPBES. (2016). *The Methodological Assessment Report on Scenarios and Models of Biodiversity and Ecosystem Services.* ecretariat of the Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services. Bonn, Germany. Retrieved March 4, 2022, from https://ipbes.net/sites/default/files/downloads/pdf/2016.methodological_assessment_report_scenarios_models.pdf

Janssen, M. A., Alessa, L. N., Barton, M., Bergin, S., & Lee, A. (2008). Towards a Community Framework for Agent-Based Modelling. *Journal of Artificial Societies and Social Simulation*, *11*(26), 1–13. https://www.jasss.org/11/2/6/6.pdf

Janssen, M. A., Pritchard, C., & Lee, A. (2020). On code sharing and model documentation of published individual and agent-based models. *Environmental Modelling & Software*, *134*, 104873. https://doi.org/10.1016/j.envsoft.2020.104873

Janssen, S., Athanasiadis, I. N., Bezlepkina, I., Knapen, R., Li, H., Domínguez, I. P., Rizzoli, A. E., & van Ittersum, M. K. (2011). Linking models for assessing agricultural land use change. *Computers and Electronics in Agriculture*, *76*(2), 148–160. https://doi.org/10.1016/j.compag.2010.10.011

Jetz, W., McGeoch, M. A., Guralnick, R., Ferrier, S., Beck, J., Costello, M. J., Fernandez, M., Geller, G. N., Keil, P., Merow, C., Meyer, C., Muller-Karger, F. E., Pereira, H. M., Regan, E. C., Schmeller, D. S., & Turak, E. (2019). Essential biodiversity variables for mapping and monitoring species populations. *Nature Ecology & Evolution*, *3*(4), 539–551. https://doi.org/10.1038/s41559-019-0826-1

Johanson, A., & Hasselbring, W. (2018). Software Engineering for Computational Science: Past, Present, Future. *Computing in Science & Engineering*, *20*(2), 90–109. https://doi.org/10.1109/MCSE.2018.021651343

Kim, H., Rosa, I. M. D., Alkemade, R., Leadley, P., Hurtt, G., Popp, A., van Vuuren, D. P., Anthoni, P., Arneth, A., Baisero, D., Caton, E., Chaplin-Kramer, R., Chini, L., De Palma, A., Di Fulvio, F., Di Marco, M., Espinoza, F., Ferrier, S., Fujimori, S., … Pereira, H. M. (2018). A protocol for an intercomparison of biodiversity and ecosystem services models using harmonized land-use and climate scenarios. *Geoscientific Model Development*. https://doi.org/10.5194/gmd-2018-115

Knapen, R., Janssen, S., Roosenschoon, O., Verweij, P., de Winter, W., Uiterwijk, M., & Wien, J.-E. (2013). Evaluating OpenMI as a model integration platform across disciplines. *Environmental Modelling & Software*, *39*, 274–282. https://doi.org/10.1016/j.envsoft.2012.06.011

Köhler, P., & Huth, A. (1998). The effects of tree species grouping in tropical rainforest modelling: Simulations with the individual-based model Formind. *Ecological Modelling*, *109*(3), 301–321. https://doi.org/10.1016/S0304-3800(98)00066-0

Lange, M., Müller, S., Fischer, T., König, S., Rojas, J. J. L., Kelbling, M., Thober, S., & Attinger, S. (2023). *FINAM* (Version 0.4). Leipzig, Germany. Retrieved June 12, 2023, from https://finam.pages.ufz.de/

Le, Q. B., Park, S. J., Vlek, P. L., & Cremers, A. B. (2008). Land-Use Dynamic Simulator (LUDAS): A multi-agent system model for simulating spatio-temporal dynamics of coupled human–landscape system. I. Structure and theoretical specification. *Ecological Informatics*, *3*(2), 135–153. https://doi.org/10.1016/j.ecoinf.2008.04.003

Lee, B. D. (2018). Ten simple rules for documenting scientific software. *PLOS Computational Biology*, *14*(12), e1006561. https://doi.org/10.1371/journal.pcbi.1006561

Lee, J.-Y., Marotzke, J., Bala, G., Cao, L., Corti, S., Dunne, J., Engelbrecht, F., Fischer, E., Fyfe, J., Jones, C., Maycock, A., Mutemi, J., Ndiaye, O., Panickal, S., & Zhou, T. (2021). Future Global Climate: Scenario-based Projections and Near-term Information. In V. Masson-Delmotte, P. Zhai, A. Pirani, S. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J. Matthews, T. Maycock, T. Waterfield, O. Yelekci, R. Yu & B. Zhou (Eds.), *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* (pp. 553–672). Cambridge University Press. https://doi.org/10.1017/9781009157896.006.

Lehmann, S., & Huth, A. (2015). Fast calibration of a dynamic vegetation model with minimum observation data. *Ecological Modelling*, *301*, 98–105. https://doi.org/10.1016/j.ecolmodel.2015.01.013

Leidinger, L., Vedder, D., & Cabral, J. S. (2021). Temporal environmental variation may impose differential selection on both genomic and ecological traits. *Oikos*, *130*(7), 1100–1115. https://doi.org/10.1111/oik.08172

Lippe, M., Bithell, M., Gotts, N., Natalini, D., Barbrook-Johnson, P., Giupponi, C., Hallier, M., Hofstede, G. J., Le Page, C., Matthews, R. B., Schlüter, M., Smith, P., Teglio, A., & Thellmann, K. (2019). Using agent-based modelling to simulate social-ecological systems across scales. *GeoInformatica*, *23*(2), 269–298. https://doi.org/10.1007/s10707-018-00337-8

Lorscheid, I., & Meyer, M. (2016). Divide and conquer: Configuring submodels for valid and efficient analyses of complex simulation models. *Ecological Modelling*, *326*, 152–161. https://doi.org/10.1016/j.ecolmodel.2015.11.013

Maji, A. K., Gorenstein, L., & Lentner, G. (2020). Demystifying Python Package Installation with conda-env-mod. *2020 IEEE/ACM International Workshop on HPC User Support Tools (HUST) and Workshop on Programming and Performance Visualization Tools (ProTools)*, 27–37. https://doi.org/10.1109/HUSTProtools51951.2020.00011

Malawska, A., & Topping, C. J. (2018). Applying a biocomplexity approach to modelling farmer decision-making and land use impacts on wildlife (J. du Toit, Ed.). *Journal of Applied Ecology*, *55*(3), 1445–1455. https://doi.org/10.1111/1365-2664.13024

Malawska, A., Topping, C. J., & Nielsen, H. Ø. (2014). Why do we need to integrate farmer decision making and wildlife models for policy evaluation? *Land Use Policy*, *38*, 732–740. https://doi.org/10.1016/j.landusepol.2013.10.025

Malchow, A.-K., Bocedi, G., Palmer, S. C. F., Travis, J. M. J., & Zurell, D. (2021). RangeShiftR: An R package for individual-based simulation of spatial eco-evolutionary dynamics and species' responses to environmental changes. *Ecography*, *44*(10), 1443–1452. https://doi.org/10.1111/ecog.05689

Manabe, S. (2019). Role of greenhouse gas in climate change. *Tellus A: Dynamic Meteorology and Oceanography*, *71*(1), 1620078. https://doi.org/10.1080/16000870.2019.1620078

Martin, R. C. (Ed.). (2009). *Clean code: A handbook of agile software craftsmanship.* Prentice Hall.

Marwick, B., Boettiger, C., & Mullen, L. (2018). Packaging Data Analytical Work Reproducibly Using R (and Friends). *The American Statistician*, *72*(1), 80–88. https://doi.org/10.1080/00031305.2017.1375986

McConnell, S. (2004). *Code Complete* (2nd ed). Microsoft Press.

McIntire, E. J. B., Chubaty, A. M., Cumming, S. G., Andison, D., Barros, C., Boisvenue, C., Haché, S., Luo, Y., Micheletti, T., & Stewart, F. E. C. (2022). PERFICT: A Re-imagined foundation for predictive ecology. *Ecology Letters*, *25*(6), 1345–1351. https://doi.org/10.1111/ele.13994

Mislan, K. A. S., Heer, J. M., & White, E. P. (2016). Elevating The Status of Code in Ecology. *Trends in Ecology & Evolution*, *31*(1), 4–7. https://doi.org/10.1016/j.tree.2015.11.006

Nowogrodzki, A. (2019). Tips for Open-Source Software Support. *Nature*, *571*(7763), 133–134. https://doi.org/10.1038/d41586-019-02046-0

O'Sullivan, D., Evans, T., Manson, S., Metcalf, S., Ligmann-Zielinska, A., & Bone, C. (2016). Strategic directions for agent-based modeling: Avoiding the YAAWN syndrome. *Journal of Land Use Science*, *11*(2), 177–187. https://doi.org/10.1080/1747423X.2015.1030463

Pan, H., & Chen, Z. (2021). Crop Growth Modeling and Yield Forecasting. In L. Di & B. Üstündağ (Eds.), *Agro-geoinformatics: Theory and Practice* (pp. 205–220). Springer International Publishing. https://doi.org/10.1007/978-3-030-66387-2_11

Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., Leprevost, F. d. V., Fufezan, C., Ternent, T., Eglen, S. J., Katz, D. S., Pollard, T. J., Konovalov, A., Flight, R. M., Blin, K., & Vizcaíno, J. A. (2016). Ten Simple Rules for Taking Advantage of Git and GitHub. *PLOS Computational Biology*, *12*(7), e1004947. https://doi.org/10.1371/journal.pcbi.1004947

Pilowsky, J. A., Colwell, R. K., Rahbek, C., & Fordham, D. A. (2022). Process-explicit models reveal the structure and dynamics of biodiversity patterns. *Science Advances*, *8*(31), eabj2271. https://doi.org/10.1126/sciadv.abj2271

Piorr, A., Ungaro, F., Ciancaglini, A., Happe, K., Sahrbacher, A., Sattler, C., Uthes, S., & Zander, P. (2009). Integrated assessment of future CAP policies: Land use changes, spatial patterns and targeting. *Environmental Science & Policy*, *12*(8), 1122–1136. https://doi.org/10.1016/j.envsci.2009.01.001

Pörtner, H.-O., Scholes, R. J., Agard, J., Archer, E., Arneth, A., Bai, X., Barnes, D., Burrows, M., Chan, L., Cheung, W. L. (, Diamond, S., Donatti, C., Duarte, C., Eisenhauer, N., Foden, W., Gasalla, M. A., Handa, C., Hickler, T., Hoegh-Guldberg, O., ... Ngo, H. (2021, June 24). *Scientific outcome of the IPBES-IPCC co-sponsored workshop on biodiversity and climate change*. IPBES secretariat. Bonn, Germany. https://doi.org/10.5281/zenodo.5101125

Prabhu, P., Jablin, T. B., Raman, A., Zhang, Y., Huang, J., Kim, H., Johnson, N. P., Liu, F., Ghosh, S., Beard, S., Oh, T., Zoufaly, M., Walker, D., & August, D. I. (2011). A survey of the practice of computational science. *State of the Practice Reports*, 1–12. https://doi.org/10.1145/2063348.2063374

Pütz, S., Groeneveld, J., Alves, L., Metzger, J., & Huth, A. (2011). Fragmentation drives tropical forest fragments to early successional states: A modelling study for Brazilian Atlantic forests. *Ecological Modelling*, *222*(12), 1986–1997. https://doi.org/10.1016/j.ecolmodel.2011.03.038

Ram, K., Boettiger, C., Chamberlain, S., Ross, N., Salmon, M., & Butland, S. (2019). A Community of Practice Around Peer Review for Long-Term Research Software Sustainability. *Computing in Science Engineering*, *21*(2), 59–65. https://doi.org/10.1109/MCSE.2018.2882753

Reidsma, P., Janssen, S., Jansen, J., & van Ittersum, M. K. (2018). On the development and use of farm models for policy impact assessment in the European Union – A review. *Agricultural Systems*, *159*, 111–125. https://doi.org/10.1016/j.agsy.2017.10.012

Robinson, D. T., Di Vittorio, A., Alexander, P., Arneth, A., Barton, C. M., Brown, D. G., Kettner, A., Lemmen, C., O'Neill, B. C., Janssen, M., Pugh, T. A. M., Rabin, S. S., Rounsevell, M., Syvitski, J. P., Ullah, I., & Verburg, P. H. (2018). Modelling feedbacks between human and natural processes in the land system. *Earth System Dynamics*, *9*(2), 895–914. https://doi.org/10.5194/esd-9-895-2018

Rollins, N. D., Barton, C. M., Bergin, S., Janssen, M. A., & Lee, A. (2014). A Computational Model Library for publishing model documentation and code. *Environmental Modelling & Software*, *61*, 59–64. https://doi.org/10.1016/j.envsoft.2014.06.022

Romero-Mujalli, D., Jeltsch, F., & Tiedemann, R. (2019). Individual-based modeling of eco-evolutionary dynamics: State of the art and future directions. *Regional Environmental Change*, *19*(1), 1–12. https://doi.org/10.1007/s10113-018-1406-7

Ropella, G. E., Railsback, S. F., & Jackson, S. K. (2002). Software Engineering Considerations for Individual-Based Models. *Natural Resource Modeling*, *15*(1), 5–22. https://doi.org/10.1111/j.1939-7445.2002.tb00077.x

Rosenzweig, C., Jones, J. W., Hatfield, J. L., Ruane, A. C., Boote, K. J., Thorburn, P., Antle, J. M., Nelson, G. C., Porter, C., Janssen, S., Asseng, S., Basso, B., Ewert, F., Wallach, D., Baigorria, G., & Winter, J. M. (2013). The Agricultural Model Intercomparison and Improvement Project (AgMIP): Protocols and pilot studies. *Agricultural and Forest Meteorology*, *170*, 166–182. https://doi.org/10.1016/j.agrformet.2012.09.011

Rosenzweig, C., Arnell, N. W., Ebi, K. L., Lotze-Campen, H., Raes, F., Rapley, C., Smith, M. S., Cramer, W., Frieler, K., Reyer, C. P. O., Schewe, J., van Vuuren, D., & Warszawski, L. (2017). Assessing inter-sectoral climate change risks: The role of ISIMIP. *Environmental Research Letters*, *12*(1), 010301. https://doi.org/10.1088/1748-9326/12/1/010301

Samaniego, L., Kumar, R., & Attinger, S. (2010). Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale: MULTISCALE PARAMETER REGIONALIZATION. *Water Resources Research*, *46*(5). https://doi.org/10.1029/2008WR007327

Sanders, R., & Kelly, D. (2008). Dealing with Risk in Scientific Software Development. *IEEE Software*, *25*(4), 21–28. https://doi.org/10.1109/MS.2008.84

Scheller, R. M., Sturtevant, B. R., Gustafson, E. J., Ward, B. C., & Mladenoff, D. J. (2010). Increasing the reliability of ecological models using modern software engineering techniques. *Frontiers in Ecology and the Environment*, *8*(5), 253–260. https://doi.org/10.1890/080141

Schmidt, A., Necpalova, M., Zimmermann, A., Mann, S., Six, J., & Mack, G. (2017). Direct and Indirect Economic Incentives to Mitigate Nitrogen Surpluses: A Sen-

sitivity Analysis. *Journal of Artificial Societies and Social Simulation*, *20*(4), 77. https://doi.org/10.18564/jasss.3477
Accepted: 2019-11-05T15:53:51Z.

Schouten, R., Vesk, P. A., & Kearney, M. R. (2020). Integrating dynamic plant growth models and microclimates for species distribution modelling. *Ecological Modelling*, *435*, 109262. https://doi.org/10.1016/j.ecolmodel.2020.109262

Schreinemachers, P., & Berger, T. (2011). An agent-based simulation model of human–environment interactions in agricultural systems. *Environmental Modelling & Software*, *26*(7), 845–859. https://doi.org/10.1016/j.envsoft.2011.02.004

Shrestha, N. K., Leta, O. T., De Fraine, B., Garcia-Armisen, T., Ouattara, N. K., Servais, P., van Griensven, A., & Bauwens, W. (2013). Modelling Escherichia coli dynamics in the river Zenne (Belgium) using an OpenMI based integrated model. *Journal of Hydroinformatics*, *16*(2), 354–374. https://doi.org/10.2166/hydro.2013.171

Sibly, R. M., Grimm, V., Martin, B. T., Johnston, A. S. A., Kułakowska, K., Topping, C. J., Calow, P., Nabe-Nielsen, J., Thorbek, P., & DeAngelis, D. L. (2013). Representing the acquisition and use of energy by individuals in agent-based models of animal populations. *Methods in Ecology and Evolution*, *4*(2), 151–161. https://doi.org/10.1111/2041-210x.12002

Sieger, C. S., & Hovestadt, T. (2021). The effect of landscape structure on the evolution of two alternative dispersal strategies. *Ecological Processes*, *10*(1), 73. https://doi.org/10.1186/s13717-021-00343-z

Simon, R. N., & Fortin, D. (2020). Crop raiders in an ecological trap: Optimal foraging individual-based modeling quantifies the effect of alternate crops. *Ecological Applications*, *30*(5), e02111. https://doi.org/10.1002/eap.2111

Stillman, R. A., Wood, K. A., & Goss-Custard, J. D. (2016). Deriving simple predictions from complex models to support environmental decision-making. *Ecological Modelling*, *326*, 134–141. https://doi.org/10.1016/j.ecolmodel.2015.04.014

Sun, Z., Lorscheid, I., Millington, J. D., Lauf, S., Magliocca, N. R., Groeneveld, J., Balbi, S., Nolzen, H., Müller, B., Schulze, J., & Buchmann, C. M. (2016). Simple or complicated agent-based models? A complicated issue. *Environmental Modelling & Software*, *86*, 56–67. https://doi.org/10.1016/j.envsoft.2016.09.006

Synes, N. W., Brown, C., Palmer, S. C. F., Bocedi, G., Osborne, P. E., Watts, K., Franklin, J., & Travis, J. M. J. (2019). Coupled land use and ecological models reveal emergence and feedbacks in socio-ecological systems. *Ecography*, *42*(4), 814–825. https://doi.org/10.1111/ecog.04039

Theurich, G., DeLuca, C., Campbell, T., Liu, F., Saint, K., Vertenstein, M., Chen, J., Oehmke, R., Doyle, J., Whitcomb, T., Wallcraft, A., Iredell, M., Black, T., Silva, A. M. D., Clune, T., Ferraro, R., Li, P., Kelley, M., Aleinov, I., … Dunlap, R. (2016). The Earth System Prediction Suite: Toward a Coordinated U.S. Modeling Capability. *Bulletin of the American Meteorological Society*, *97*(7), 1229–1247. https://doi.org/10.1175/BAMS-D-14-00164.1

Topping, C. J., Hansen, T. S., Jensen, T. S., Jepsen, J. U., Nikolajsen, F., & Odderskær, P. (2003). ALMaSS, an agent-based model for animals in temperate European landscapes. *Ecological Modelling*, *167*(1), 65–82. https://doi.org/10.1016/S0304-3800(03)00173-X

Topping, C. J. (2011). Evaluation of wildlife management through organic farming. *Ecological Engineering*, *37*(12), 2009–2017. https://doi.org/10.1016/j.ecoleng.2011.08.010

Topping, C. J., Alrøe, H. F., Farrell, K. N., & Grimm, V. (2015). Per Aspera ad Astra: Through Complex Population Modeling to Predictive Theory. *The American Naturalist*, *186*(5), 669–674. https://doi.org/10.1086/683181

Topping, C. J. (2022). *ALMaSS - the animal, landscape and man simulation system*. Aarhus, Denmark. https://gitlab.com/ChrisTopping/ALMaSS_all

Urban, M. C., Bocedi, G., Hendry, A. P., Mihoub, J.-B., Pe'er, G., Singer, A., Bridle, J. R., Crozier, L. G., De Meester, L., Godsoe, W., Gonzalez, A., Hellmann, J. J., Holt, R. D., Huth, A., Johst, K., Krug, C. B., Leadley, P. W., Palmer, S. C. F., Pantel, J. H., … Travis, J. M. J. (2016). Improving the forecast for biodiversity under climate change. *Science*, *353*(6304), aad8466. https://doi.org/10.1126/science.aad8466

Urban, M. C., Travis, J. M. J., Zurell, D., Thompson, P. L., Synes, N. W., Scarpa, A., Peres-Neto, P. R., Malchow, A.-K., James, P. M. A., Gravel, D., De Meester, L., Brown, C., Bocedi, G., Albert, C. H., Gonzalez, A., & Hendry, A. P. (2022). Coding for Life: Designing a Platform for Projecting and Protecting Global Biodiversity. *BioScience*, *72*(1), 91–104. https://doi.org/10.1093/biosci/biab099

Vable, A. M., Diehl, S. F., & Glymour, M. M. (2021). Code Review as a Simple Trick to Enhance Reproducibility, Accelerate Learning, and Improve the Quality of Your Team's Research. *American Journal of Epidemiology*, *190*(10), 2172–2177. https://doi.org/10.1093/aje/kwab092

van Ittersum, M. K., Ewert, F., Heckelei, T., Wery, J., Alkan Olsson, J., Andersen, E., Bezlepkina, I., Brouwer, F., Donatelli, M., Flichman, G., Olsson, L., Rizzoli, A. E., van der Wal, T., Wien, J. E., & Wolf, J. (2008). Integrated assessment of agricultural systems – A component-based framework for the European Union (SEAMLESS). *Agricultural Systems*, *96*(1-3), 150–165. https://doi.org/10.1016/j.agsy.2007.07.009

Vedder, D., Ankenbrand, M., & Cabral, J. S. (2021). Dealing with software complexity in individual-based models. *Methods in Ecology and Evolution*, *12*(12), 2324–2333. https://doi.org/10.1111/2041-210X.13716

Vedder, D., Leidinger, L., & Sarmento Cabral, J. (2021). Propagule pressure and an invasion syndrome determine invasion success in a plant community model. *Ecology and Evolution*, *11*(23), 17106–17116. https://doi.org/10.1002/ece3.8348

Vedder, D., Lens, L., Martin, C. A., Pellikka, P., Adhikari, H., Heiskanen, J., Engler, J. O., & Sarmento Cabral, J. (2022). Hybridization may aid evolutionary rescue of an endangered East African passerine. *Evolutionary Applications*, *15*(7), 1177–1188. https://doi.org/10.1111/eva.13440

Vedder, D., Rakowski, J., Kolb, L., Theilen, G., Lakner, S., & Pe'er, G. (submitted). Agent-based modelling of agricultural social-ecological systems neglects biodiversity.

Will, M., Dressler, G., Kreuer, D., Thulke, H.-H., Grêt-Regamey, A., & Müller, B. (2021). How to make socio-environmental modelling more useful to support policy and management? *People and Nature*, *00*, 1–13. https://doi.org/10.1002/pan3.10207

Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., Haddock, S. H. D., Huff, K. D., Mitchell, I. M., Plumbley, M. D., Waugh, B., White, E. P., & Wilson, P. (2014). Best Practices for Scientific Computing. *PLoS Biology*, *12*(1), 1–7. https://doi.org/10.1371/journal.pbio.1001745