National Marine Fisheries Service
NOAA

Fishery Bulletin
established in 1881

Spencer F. Baird
First U.S. Commissioner of Fisheries and founder of *Fishery Bulletin*

Engineering a prototype for a next-generation stock assessment modeling platform

Jon Brodziak (contact author)[1]
Matthew Supernaw[2]
Bai Li[3]
Christine Stawitz[4]
Haley Oleynik[5]
Teresa A'mar[6]

Email address for contact author: jon.brodziak@noaa.gov

[1] Pacific Islands Fisheries Science Center
National Marine Fisheries Service, NOAA
1845 Wasp Boulevard
Honolulu, Hawaii 96818

[2] Office of Science and Technology
National Marine Fisheries Service, NOAA
263 13th Avenue South
St. Petersburg, Florida 33701

[3] ECS Federal LLC
for Office of Science and Technology
National Marine Fisheries Service, NOAA
1315 East West Highway
Silver Spring, Maryland 20910

[4] Office of Science and Technology
National Marine Fisheries Service, NOAA
7600 Sand Point Way
Seattle, Washington 98115

[5] Institute for the Oceans and Fisheries
University of British Columbia
2202 Main Mall
Vancouver, British Columbia V6T 1Z4, Canada

[6] Dragonfly Data Science
P.O. Box 27535
Wellington 6141, New Zealand

Abstract—Environmental and anthropogenic processes have led to widespread changes in productivities and spatial distributions of marine fishery resources. As geographic distributions of fish stocks and subsequently fishing fleets shift, improved spatiotemporal data and spatial modeling will become necessary to estimate abundance and productivity. The goal of this paper is to propose a strategy for engineering a modeling platform for spatially explicit, next-generation stock assessment models. We recount our approach for developing a system prototype, the Metapopulation Assessment System (MAS), that is easy to use, modular, and extensible. The MAS prototype was designed to support complex metapopulation modeling, which includes handling multiple populations, areas, fleets, and surveys and sex differentiation. We describe the components of the software life cycle, engineering, and infrastructure design to support a spatially explicit, next-generation stock assessment system. Software infrastructure was designed and implemented with 3 components: GitHub for collaboration, version control, and code organization; the C++ language for coding fishery system dynamics and estimation modeling; and the R language for coding the input-output interface and systematic testing. Systematic testing was a key component of the MAS development life cycle and was applied to guarantee that system requirements, such as accurate estimation of quantities of interest, are successfully implemented. We conclude with a discussion of some lessons learned and future challenges for ongoing efforts to implement a next-generation stock assessment platform.

Manuscript submitted 19 June 2024.
Manuscript accepted 14 January 2025.
Fish. Bull. 123:88–104 (2025).
Online publication date: 3 February 2025.
doi: 10.7755/FB.123.2.2

Environmental and anthropogenic factors increasingly affect the spatial distribution and productivities of marine fishery resources (Perry et al., 2005; Cheung et al., 2013; Free et al., 2019; Pinsky et al., 2020). As geographic distributions of fish stocks and fishing fleets shift (Gullestad et al., 2020; Palacios-Abrantes et al., 2023), improved spatiotemporal data (e.g., Maureaud et al., 2021) and spatial modeling (e.g., Koenigstein et al., 2016) will be needed to accurately estimate abundance and to predict changes in productivity. Some single-region stock assessments will need to be reconfigured as spatially explicit models or approximated as spatially implicit models to account for range shifts. Single-region assessment models can be extended to implicitly account for spatial patterning in selectivity among fishing fleets by using the "areas-as-fleets" modeling approach (e.g., Waterhouse et al., 2014). Although assessment approaches have continually evolved to improve population dynamics modeling, a next-generation platform is needed to handle the challenges of producing accurate stock assessments for populations exposed to regional changes in environmental conditions.

Improvements in modeling movements of fish and fishing fleets will be an important component of a next-generation stock assessment modeling platform that accounts for spatiotemporal effects of fisheries and environmental drivers under climate change (Berger et al., 2017; Punt, 2019). This platform will also need to support management strategy evaluations with spatially structured operating models (OMs), ensemble modeling, and ecosystem-based fishery management applications (Lynch et al., 2018). A next-generation modeling system will require a modular and extensible software infrastructure to handle changes in the data types used for assessment purposes, analytical approaches, and nonstationary environmental conditions.

Our goal with this paper is to propose a strategy for engineering a modeling system for spatially explicit, next-generation stock assessment models. To this end, we recount our development and testing of a prototype modeling platform for building spatially explicit assessment models that is modular, extensible, and easy to use. We describe the components of the software life cycle, engineering, and infrastructure design needed to support a spatially explicit, next-generation stock assessment modeling platform. Software infrastructure was designed and implemented with 3 components: GitHub for collaboration, version control, and code organization; the C++ language for coding fishery system dynamics and estimation modeling; and the R language for coding the input-output interface and systematic testing. We explain how a prototype, the Metapopulation Assessment System (MAS), was developed to meet these requirements. Systematic testing was a key component of the MAS development life cycle and was applied to guarantee that system requirements, such as accurate estimation of quantities of interest, are successfully implemented. We illustrate an application of simulation testing with the MAS estimation model (EM) for an age-structured OM. We discuss some lessons learned from the MAS project and future challenges for ongoing efforts to implement a next-generation stock assessment modeling platform.

Materials and methods

Engineering a next-generation modeling system is a key goal of this effort to support fishery assessments and management. In 2014, we initiated the development of the MAS. The MAS was conceptualized and developed as an informal collaboration between the NOAA Pacific Islands and Southeast Fisheries Science Centers (Brodziak et al., 2014, 2017). The vision for the MAS project was to build a spatial prototype to span future modeling needs while maintaining existing capabilities with an object-oriented programming (OOP) approach. This aim was consistent with NOAA's updated Stock Assessment Improvement Plan (Lynch et al., 2018), which outlined a future with improved continuity between regional stock assessment methods under a generalized stock assessment framework. It was also consistent with recommendations from the Center for the Advancement of Population Assessment Methodology workshop on next-generation stock assessment models and their requirements held in 2019 (Hoyle et al.[1]; Punt et al., 2020).

The MAS was conceived and designed to be a general software system to support an interoperable set of software modules for building stock assessment models. The main design features of the MAS are a modular and extensible software infrastructure engineered for flexibility and a development life cycle for maintainability and quality control. The MAS development strategy was to support alternative assessment model types with templates as well as to provide built-in ensemble modeling capabilities to handle parametric and structural uncertainty under the system life cycle.

System life cycle, engineering, and infrastructure

System life cycle The system life cycle supports a rigorous software engineering process by using "a systematic approach to the analysis, design, assessment, implementation, test, maintenance and reengineering of software" (Laplante, 2001). A system life cycle and quality control procedures are essential for engineering a next-generation stock assessment modeling platform.

Developing software to build an assessment modeling platform requires a total quality assurance and improvement cycle. The system life cycle is a sequence of 4 primary processes of software requirements, design, implementation, and testing. Each process is represented as a node in a directed graph that spans the set of one-step interactions among processes in the life cycle.

When fishery system dynamics change, either because of changes in our understanding of the system with increased data availability or in response to novel environmental conditions, the assessment modeling approach must also adapt. The design components of the software modeling system evolve in the development life cycle. The first step in the life cycle is to set requirements. Specifying the necessary software behaviors and attributes to implement system requirements conditioned on resource constraints is an iterative process. Technical requirements for a next-generation model were discussed and reviewed at the Center for the Advancement of Population Assessment Methodology workshop in 2019 (Hoyle et al.[1]) and summarized by Punt et al. (2020). These requirements include clear and thorough documentation, state-of-the-art diagnostic analyses, objective means to perform data weighting and model tuning, inherent capacity to account for spatial structure with stock components, capacity to include tagging observations including close-kin mark-recapture data, state-space modeling capacity with flexible implementation of random effects, and support for future stock projections and management strategy evaluations. The next-generation modeling system should support these technical requirements as well as be able to be used to implement ensemble modeling that directly accounts for structural uncertainty (e.g., Dormann et al., 2018; Jardim et al., 2021).

The design process is the second step and is used to identify how to meet requirements and satisfy constraints by evaluating alternative solutions and choosing the best available ones. Identifying the design tools and infrastructure needed to build the software follows the requirements. The implementation process sets the timing pattern of development tasks and their prioritization to manage potential risks to success. Developing efficient processes to precisely implement the software system requirements and build the design components is the third step of the life cycle.

[1] Hoyle, S. D., M. N. Maunder, and Z. T. A'mar. 2020. Frameworks for the next generation of general stock assessment models: 2019 CAPAM workshop report. N.Z. Fish. Assess. Rep. 2020/39, 33 p. Minist. Prim. Ind., Wellington, New Zealand. [Available from website.]

Quality assurance of stock assessments is inherently difficult because the exact estimated quantities and assessment results are nearly impossible to know. It is, however, critical to have a strong testing process that can at least address the performance of models. The testing process is the last step in the cycle and is used to set the methods for quality control and assurance for assessment software, in order to measure progress and to adjust other processes to achieve success. Unit and simulation testing procedures for reliably verifying software features for ongoing quality assurance and control are described in the "Systematic testing" section (see also Supplementary Materials). Taken together, the 4 steps of requirements, design, implementation, and testing are analogous to those of the Deming cycle used to achieve total quality improvement in manufacturing (Deming, 1982). Typically, the workflow for any production process, including software engineering, entails feedback. Such feedback can occur between pairs of life cycle processes. Accounting for feedback, the life cycle functions as a means to iteratively plan, implement, test, and rethink the interrelated processes needed to build the next-generation system. Overall, the conceptual design of the MAS is based on a system life cycle that focuses work on software engineering and infrastructure to achieve ongoing quality assurance and improvement.

Software engineering Given the system life cycle, it is important to emphasize the necessity of applying software engineering principles (e.g., Martin, 2009) in a systematic and measured approach to the development and maintenance of software. In the case of our development of a modeling system, the overall purpose was to structure the computer code needed to build a next-generation assessment platform. Under a systematic approach, one begins by analyzing the problem and breaking it down into smaller components, or modules. Modularity allows development of well-defined, independent components that have logically discrete functionality. The 3 basic modules in the MAS, for example, are *CalculateCatchAtAge()*, *CalculateSpawningBiomass()*, and *SetVarianceCovariance()*; these modules can be used to produce predictions of fishery catch at age and population spawning biomass and to set estimates of parameter covariances to their estimated values, respectively.

Each module is implemented and tested in isolation prior to integration. Unit testing can be used to write single test cases to confirm the accuracy of individual program modules. Modules also accommodate division of work and ultimately improve maintainability. This modular approach, in turn, leads to an improved capacity for extensibility, which is another key design feature of the MAS. Also, one of the challenges is to allow the flexibility of modular programming while also making the code run efficiently. With modular programming in the MAS, functional forms become object functors, all inherited from a base class with a virtual function that the child classes override. This eliminates the need for costly conditional statements that result in jumps in the machine code. In addition, multidimensional arrays are flattened to 1-dimensional arrays. This array flattening keeps memory contiguous and aids in the efficiency of the central processing unit cache, although the effects on runtime were not explicitly evaluated for this study.

Modularity can be applied in many places to help extend a generalized stock assessment framework to include new features for alternative model configurations. These model extensions include using alternative sub-models for life history characteristics, adding new observational data types, deploying alternative input and output options, applying different estimation methods, and changing the spatial assessment structure, to name a few. The MAS can handle such future changes because it is programmed by using the C++11 (ISO/IEC 14882:2011) and R (vers. 4.1.2; R Core Team, 2021) languages that provide for flexible and extensible software engineering.

Adhering to a structured software engineering approach also entails the consistent application of 2 important software paradigms, functional programming and OOP. In the MAS prototype, both functional programming and OOP are used to take advantage of their complementary natures. In functional programming, computer program structure is divided into individual modules that are small, encode one functional operation, and are simple to describe. This divide-and-conquer approach to coding facilitates understanding among developers and allows straightforward modification of existing code. Functional programming is a top-down design approach that leads to a software system that is logically divided into modular parts.

In comparison to functional programming, OOP is a software approach in which a model is constructed around objects. Under the OOP paradigm, data are compartmentalized into objects (i.e., data fields or attributes), and the contents and behavior of objects are defined through the declaration of classes (i.e., methods). The primary goal of OOP is to increase the flexibility and maintainability of code by organizing it around objects that represent real-world or conceptual entities. In theory, an object is a software representation of a real-world object. In practice, an object is a runtime state or instance of a class, where a class is a compile-time structure that comprises data and associated operations. Just as with real-world objects of modeling interest like fish or fishing fleets, software objects have state and behavior. For example, an oceanic fish has a state (species, morphological characteristics, and genetic signature) and behavior (feeding, spawning, and avoiding predation).

In the MAS, classes are defined for a variety of assessment system components, including structures for population dynamics, modeling information, spatial data, objective functions, thread management (the coordination and control of multiple independent sequences of execution to optimize performance within a program), output processes, and others. These classes are used to implement system-level concepts that relate object structures and their behaviors in an efficient manner. The MAS is one of a handful of fishery analysis platforms designed to embrace OOP and modular design for software infrastructure. Other platforms that use OOP design features include the FLR framework (Kell et al., 2007; Fisheries Library in R, available from website), Data-limited Methods Toolkit (Carruthers and Hordyk, 2018; DLMtool, available from website), OSMOSE model (Shin and Cury, 2004; Object-oriented Simulator of Marine Ecosystems, available from website), and Casal2 (Doonan et al., 2016; available from website).

Software infrastructure The development of the MAS is based on a flexible and efficient software infrastructure. Initially, Git (available from website) was used for software version control, and the MAS project was deployed on GitHub. We used the GitHub application system (available from website) for version control, collaboration, hosting, and code distribution. Infrastructure features were important for organizing software development and facilitated the structured construction of the MAS prototype and the associated R package r4MAS (Supernaw et al., 2022).

Our general strategy for building the software infrastructure of the MAS was based on 4 principles of OOP: encapsulation, data abstraction, polymorphism, and inheritance. Encapsulation refers to an object's ability to hide its data attributes and behavior that are not necessary for its user or client process. Encapsulation allows an object's data members and methods to be represented as a single unit (e.g., a Kuan) with protection from client modification. The data (attributes) and the methods (functions or procedures) that operate on the data are bundled into a single unit or class. This concept helps in hiding the internal state of the object and in exposing only a controlled interface. The benefits of encapsulation are protection of data from accidental corruption as well as reduction of the coupling between code fragments. This access control, in turn, provides safeguards for data processing and improves code maintainability.

Data abstraction is the process of hiding the complex software implementation details and showing only the essential features of the object. It helps in reducing complexity and allows a programmer to focus on interactions at a high level. In a fishery stock assessment model, parameters such as growth rates or mortality can be refit by using a bootstrap procedure on the input data, a procedure that resamples the observed data and refits the model to generate new estimates of model parameters. This process can be encapsulated in a method like *refitModel()*, where the user triggers the model fitting procedure with bootstrapped data without needing to manage the underlying resampling logic. Data abstraction is achieved through encapsulation. By using encapsulation, the internal implementation details of a class can be hidden and only the necessary parts are exposed (through public methods). In general, data abstraction simplifies interactions with complex models by providing high-level operations that hide intricate computational processes.

Polymorphism refers to the ability to implement abstract programming entities in multiple ways to extend software features. In the context of OOP, polymorphism is the capacity to provide different kinds of functionality with the same interface. Polymorphism is implemented in the MAS by using template metaprogramming. An example of polymorphism is illustrated in Pseudocode Block 1, with 2 methods of *Evaluate* returning different outputs depending on the type of selectivity class selected by the

Pseudocode Block 1

```
Class SelectivityBase{
public:
      virtual double Evaluate(double age)=0;
};

class LogisticSelectivity : SelectivityBase{
      double a50; //age of 50% selectivity
      double s; //the rate of increase in selectivity at a50

virtual double Evaluate(double age) {
  return (1.0) / ((1.0) + exp(-1.0 + (age - a50) /s))/max_selectivity;
  }
};

class DoubleLogisticSelectivity : SelectivityBase {
      double a50; //age of 50% selectivity
      double alpha_asc; //ascending alpha
      double beta_asc; // ascending beta
      double alpha_desc; // descending alpha
      double beta_desc; // descending beta

virtual double Evaluate(double age) {
  variable a_= 1.0 / (1.0 + exp(-1.0 * (age -alpha_asc)/ beta_asc));
  variable b_ = 1.0 - (1.0 / (1.0 + exp{-1.0 * (age - alpha_desc) / beta_desc)));
  return (a_* b_)/max_selectivity;
```

user, either the logistic (2-parameter model) or the double-logistic (4-parameter model) form of fishery selectivity. This example also demonstrates how polymorphism can streamline the syntax of model specification. Note that, for brevity, the code for calculating *max_selectivity*, the maximum value of unscaled selectivity at age, is not provided in Pseudocode Block 1.

Inheritance in OOP allows new object definitions (children) to be based on existing ones (parents), such that "when a new kind of object class is defined, only those properties that differ from the properties of the specified existing classes need to be declared explicitly, while the other properties are automatically extracted from the existing classes and included in the new class" (Taivalsaari, 1996). An example can be taken from taxonomy. If a parent class *Fish* is defined, a property shared among all fish species (e.g., HasGills and IsAquatic) need not be redefined for the child class *CartilaginousFish* because it inherits these properties from its parent; however, definitions would be needed for properties of cartilaginous fishes (e.g., IsCartilaginous and SwimBladder=FALSE) that are not shared with bony fishes. Through inheritance, the definition and implementation of a new class can be derived from an existing base class, with the newly derived class inheriting attributes from the base class.

Inheritance provides an efficient way to extend the software infrastructure needed to create models with alternative sub-model components. For example, the parent class *SelectivityBase* in Pseudocode Block 1 has 2 child classes, *LogisticSelectivity* and *DoubleLogisticSelectivity*, that inherit the method *Evaluate* for calculation of selectivity values. The inheritance structure used for population dynamics in the MAS prototype (Figs. 1 and 2) includes base classes for fish recruitment, growth, mortality, maturity, selectivity, and a likelihood-based observation model. The child subclasses for modeling population dynamics for growth, recruitment, mortality, maturity, and selectivity have 6, 6, 2, 2, and 4 subclasses, respectively (Figs. 1 and 2). Similarly, the parent class for creating a joint negative log-likelihood function (*Negative-Loglikelihood*) has 5 subclasses corresponding to alternative probability distributions for likelihood components (Fig. 2). Overall, the structured nature of functional programming and the encapsulation, data abstraction, polymorphism, and inheritance aspects of OOP provide powerful tools for building a next-generation assessment platform.

The C++ language is the computational engine for the MAS, a next-generation stock assessment system. The MAS is coded in templated C++. The code is written in pure standard C++11 and is, therefore, portable across operating systems. The MAS is currently compiled with the Analytics Template Library (ATL, available from website) and can also be compiled with Template Model Builder (Kristensen et al., 2016; TMB, available from website). The ATL has several quasi-Newton minimizers available in its template library, and the TMB has R-based minimizers. The MAS is not limited to just these template libraries, and any C++-based optimization library could be used. When the R package r4MAS is used, an objective function can be minimized with any available R minimizer. As for output, the standard function value, parameter values, gradient, and Hessian matrix are captured along with all derived quantities and exported in JavaScript Object Notation or JSON format, which is easily parsed into a list object for output processing in R.

In the MAS, C++ is also used to accommodate flexible OOP structuring (Jana, 2005; Stroustrup, 2013) and to support template metaprogramming. Template metaprogramming is the capacity to construct functions or classes that have the capacity to operate on multiple different object types (e.g., Vandevoorde et al., 2018). Function templates, which are functions that are parameterized so that they represent a family of functions, or other template types, are used to achieve a plug-and-play modularity implemented in C++ code. These templates are based on the concept of functors, or a class object that can be called like a function, providing flexibility in the application of parameter optimization routines. For example, the minimum value functor *ad_min()* is as follows:

*template <typename T> inline const atl::Variable<T> ad_min(const atl::Variable<T>& a, const atl::Variable<T>& b, atl::Variable<T> C = 1e-5) { return (a + b - atl::ad_fabs(a - b,C))*0.5; }*

This functor in the MAS has 3 arguments: the arguments *a* and *b* are the data pair being compared to find their minimum value, and the argument *C* is a small constant used for smoothing.

The functor *ad_min()* eliminates the need for a conditional *if* branch to calculate the minimum value. The lack of a branching statement in *ad_min()*, in turn, allows application of automatic differentiation algorithms (Griewank and Walther, 2008). Automatic differentiation cannot be used to optimize objective functions that include branching statements. This feature of branching avoidance is part of the MAS when used with the ATL or other automatic differentiation systems (e.g., CppAD, available from website). Another benefit of template metaprogramming is being able to plug in different estimation algorithms. For example, the Laplace approximation method can be applied to estimate the marginal likelihood for a state-space assessment model when it is a smooth unimodal function, but a more costly particle filtering method may be needed when the marginal likelihood is not sufficiently smooth (e.g., Auger-Méthé et al., 2021).

Plug-in flexibility is important because the next-generation platform needs to make it possible to formulate assessment models based on either frequentist (Fournier and Archibald, 1982; Methot, 1990; Legault and Restrepo, 1998) or Bayesian (Punt and Hilborn, 1997; Meyer and Millar, 1999) frameworks. This adaptability includes random-effects or mixed-model frameworks (Nielsen and Berg, 2014; Miller et al., 2016), which can be statistically formulated by using either frequentist or Bayesian estimators (e.g., Demidenko, 2004; Auger-Méthé et al., 2021). Overall, the statistical estimation framework needs to be flexible for general assessment applications.

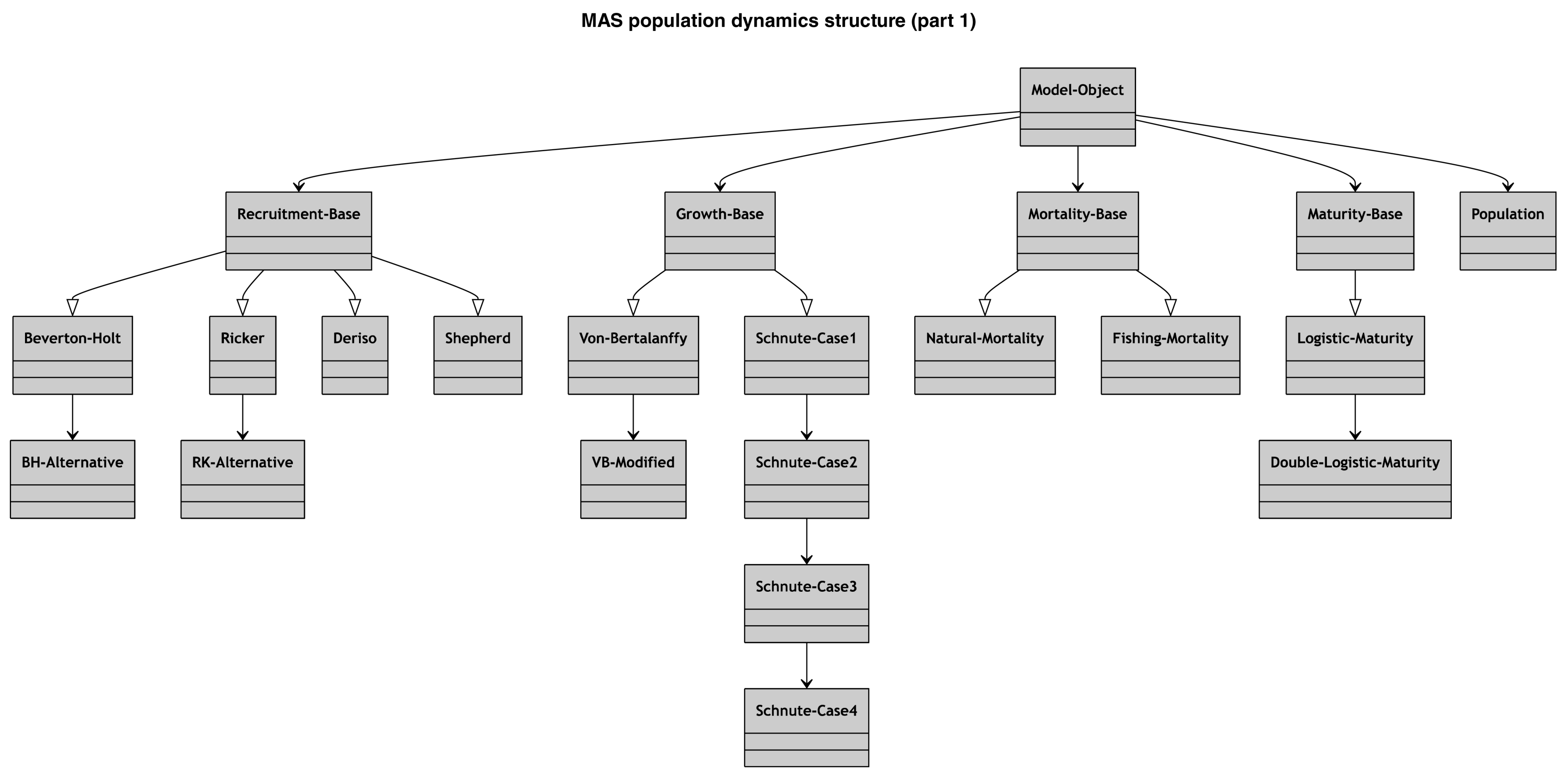

Figure 1

Schematic of some of the parent classes and inheritance structure for population dynamics in the Metapopulation Assessment System (MAS), a next-generation stock assessment modeling platform, including the base classes of Recruitment, Growth, Mortality, Maturity, and Population, along with their child subclasses. For a schematic with the other base classes, Selectivity, Negative-Loglikelihood, Fleet, Survey, Movement, and Information, see Figure 2.

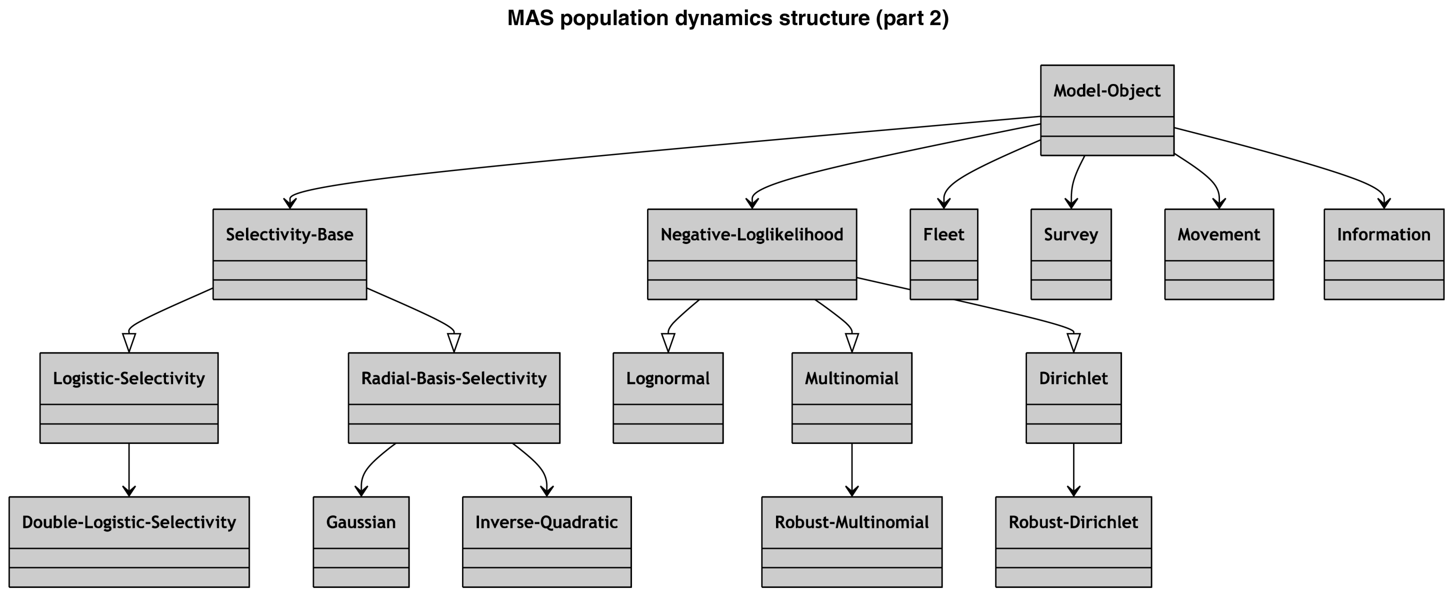

Figure 2

Schematic of some of the parent classes and inheritance structure for population dynamics in the Metapopulation Assessment System (MAS), a next-generation stock assessment modeling platform, including the base classes of Selectivity, Negative-Loglikelihood, Fleet, Survey, Movement, and Information, along with their child subclasses. For a schematic with the other base classes, Recruitment, Growth, Mortality, Maturity, and Population, see Figure 1.

In the MAS, the R language and environment is used to provide an application programming interface that supports interoperability, modifiability, and portability, with numerous extensions. Preprocessing and post-processing of MAS input and output data streams can be efficiently implemented by using the R language and its numerous extensions (Wickham, 2016; Wickham and Grolemund, 2017). The R language promotes the sharing of open-source scientific computing technology because of the dynamic documentation of reproducible numerical calculations and results (Xie, 2015; Xie et al., 2019) and the construction of R packages (Wickham, 2015). The combination of a scientific calculation engine based on the C++ language that can efficiently implement OOP (Stroustrup, 2013) with an interface and calculation engines based on the R language is a key design component for statistical computing (Eubank and Kupresanin, 2011) in a next-generation stock assessment system. This integration can be accomplished with the R package Rcpp (Eddelbuettel, 2013), which is a key tool for integrating the functionality of C++ and R.

Overall, the important components of software infrastructure needed for the design and implementation of the MAS prototype included the following: 1) Git for software version control; 2) GitHub for collaboration, version control, hosting, and organizing work; 3) the C++ language for coding the population dynamics and assessment estimation modeling; 4) the R language for coding the user interface and systematic testing; 5) an optimization template library, such as the ATL; and 6) the Rcpp package to smoothly integrate the functionality of C++ and R. These software infrastructure components are essential for implementing our strategy to build and maintain a next-generation stock assessment platform.

Model uncertainty and ensemble specification

Characterizing uncertainty in assessment model estimates is an important component of a next-generation stock assessment platform. In single-model assessments, characterizing uncertainty has been well addressed with standard methods (Smith et al., 1993; Patterson et al., 2001; Privitera-Johnson and Punt, 2020). However, accounting for structural uncertainties in predictive models is often needed (Peterman, 2004; Jardim et al., 2021; Ducharme-Barth and Vincent, 2022). For example, structural uncertainty about processes that influence spatial population dynamics under climate change is nontrivial because the environmental conditions have not been observed. An ensemble modeling approach may be needed to account for and bound structural uncertainties in spatial dynamics in order to reduce bias through model averaging (e.g., Dormann et al., 2018; Brooks and Brodziak, 2024). As a result, a requirement for any next-generation modeling system is the capacity to precisely specify each component of ensemble models, including different spatial structures as needed to account for structural uncertainties. This capacity includes the functionality to build spatially explicit models with one-to-many, many-to-one, or many-to-many relationships between regions and sub-model components. Overall, any next-generation system should be able to efficiently support spatial ensemble models.

The MAS was designed to support ensemble modeling for spatially explicit assessments, and ensemble modeling may be needed to represent the real-world knowledge base and process uncertainty in stock assessments. Ensemble construction requires identifying trade-offs among alternative model representations and setting the model weights needed to combine modeling results (e.g., Burnham and Anderson, 2002; Jardim et al., 2021). Estimates from an ensemble model should account for both estimation and model structural uncertainty (e.g., Brodziak and Piner, 2010; Ducharme-Barth and Vincent, 2022). Structural uncertainty includes the choice of the spatial structure used to model a fishery system.

For implementation of features, the MAS prototype includes an ensemble engine in C++ that can be used to construct ensemble models through template metaprogramming. Given a set of structural uncertainties, the engine can be used to create the ensemble as a factorial combination of all possible instances of sub-model factor levels or structural forms. For example, if the uncertainties consisted of a growth parameter, such as asymptotic length, and the functional form of an expected stock-recruitment function, the ensemble engine would iterate over all combinations of growth parameter values and recruitment sub-models to construct the members of the ensemble. The engine assigns each ensemble member, or individual assessment model, a unique identifier based on the signature of its sub-model components, automating the ensemble generation process. Each model is run in parallel by using Message Passing Interface (Message Passing Interface Forum, available from website) to decrease evaluation time. Structural uncertainties can also include whether a stock area is represented with a single-region or multiple-subregion model, but a feature that addresses this issue is not currently implemented in the ensemble engine of r4MAS.

Systematic testing

Systematic testing for quality assurance and control is a key to reliable product development. Regular scientific software testing procedures are highly recommended for maintaining functionality and creating new features (Wilson et al., 2014, 2017). For assessment modeling, the development and application of quality assurance entails making good decisions about the types and extent of empirical testing analyses along with ensuring that coverage of the space of plausible observation and process sub-models is adequate.

One means to efficiently conduct quality control and assurance tests for the MAS has been to use a software collaboration platform like GitHub. This platform is used for quality control and assurance in the calculation engine written in C++ and for the input and output processing in the r4MAS package, the R language interface for the MAS. For example, we use unit tests to confirm the accuracy of the Beverton–Holt recruitment module. These tests involve fixing all parameters within the module with known input values and assessing the module's output. Additionally, we have unit tests that verify whether the model estimates parameters when the *estimated* flags are turned on for the Beverton–Holt recruitment module. Furthermore, we conduct integration tests that load all modules of the MAS with predefined input values, comparing the MAS estimates with the expected true values. All the tests are written in R by using the *testthat* framework (Wickham, 2011). In general, it is a recommended practice to be explicit about uncertainty and to quantify it in complex models used for scientific policy analyses (e.g., Brodziak and Link, 2002; National Research Council, 2012). Systematic testing is an important component in the development of the MAS and the r4MAS package, and we illustrate this significance with an example in the next section.

In the rest of this section, we discuss how simulation testing can be used to evaluate how the r4MAS package performs as a stock assessment EM under one iteration of the software development life cycle. In a hypothetical systematic testing iteration, we addressed a hypothetical question about the performance of r4MAS in estimating stock status relative to reference points based on maximum sustainable yield (MSY). What is the relative accuracy of estimates of stock status from the r4MAS package? To illustrate the simulation testing approach, we applied the age-structured OM developed by Li et al. (2021) to evaluate the predictive accuracy of r4MAS as an EM. We used the exact same population simulation model and testing configuration used by Li et al. (2021) to examine estimation accuracy under their scenario for a hypothetical population based on life history traits for bottomfish species in the Atlantic Ocean off the southeastern United States. Model structures and equations of both the OM and the EM are detailed in the Supplementary Materials. In what follows, we briefly describe the analysis approach of Li et al. (2021), the OM and EM, the test cases, and the performance measures for evaluating estimation performance.

Li et al. (2021) developed an age-structured OM to evaluate predictive accuracy and reliability of 4 primary age-structured stock assessment models used in the United States. The EMs for each platform were fitted to simulated data from the OM and the simulation-estimation process was iterated 100 times with time varying recruitment deviations and observation errors (Table 1) (Li et al., 2021). Their goal was to compare how well each EM could estimate key fishery management variables, such as spawning biomass and fishing mortality, relative to biological reference points under different configurations of the OM for the fishery system.

The OM was based on a standard stock assessment configuration for a simple fishery system. It consisted of an age-structured population dynamics model with a plus-group, pooled sexes, and discrete time steps corresponding to a 30-year period. Population dynamics were based on common formulations for an age-structured model of a population harvested by a single fishing fleet and monitored by a single research survey (Suppl. Materials, Suppl. Tables 1–3). Biological reference points for producing MSY and the spawning biomass and fishing mortality associated with MSY were evaluated for each iteration of the simulation-estimation process. The EM had the same model structure as the OM and was fitted to the simulated data in each iteration.

Two performance measures were used to quantify the expected estimation accuracy for quantities of interest produced by the EM conditioned on the OM assumptions and simulated data. The first performance measure was the mean over simulations (number of simulations [n]=100) of the median absolute relative error (MARE) for estimation across annual time steps (T=30). The MARE was used to measure the expected magnitude of unsigned errors for output quantities of interest Q as follows:

$$MARE = \frac{1}{T}\left[\sum_{t=1}^{T} \underset{s\in S}{median}\left\{\left|\frac{Q_{\text{MAS,s,t}} - Q_{\text{true,s,t}}}{Q_{\text{true,s,t}}}\right|\right\}\right], \quad (1)$$

where S = the set of simulations per simulation test (n=100);
$Q_{\text{MAS,s,t}}$ = the estimate of Q from the MAS in simulation s at time step t; and
$Q_{\text{true,s,t}}$ = the true value of Q in simulation s at time step t calculated from the OM.

The second performance measure was the mean of the median relative error (MRE), which provides an index of the expected directional estimation error for quantities of interest Q as follows:

$$MRE = \frac{1}{T}\left[\sum_{t=1}^{T} \underset{s\in S}{median}\left\{\frac{Q_{\text{MAS,s,t}} - Q_{\text{true,s,t}}}{Q_{\text{true,s,t}}}\right\}\right]. \quad (2)$$

These 2 measures of estimation performance are identical to measures used in Li et al. (2021), with MARE and MRE providing measures of the absolute estimation error and expected bias of the MAS estimator. We also calculated the median absolute deviation (MAD) of the MARE values to characterize their precision (Suppl. Table 4). The MAD for a data set provides a robust measure of the standard deviation of that data set. In our application of MAD, the data set $\underline{x}$ is the set of MARE values for a quantity of interest:

$$MAD(\underline{x}) = \underset{i}{median}\left(\left|x_i - \bar{x}\right|\right), \quad (3)$$

where x_i = the i^{th} MARE value; and
$\bar{x}$ = the average of the set of MARE values.

Two measures of stock status were chosen as quantities of interest for testing the estimation accuracy of the MAS. These were the relative spawning biomass, which is the ratio of spawning biomass to the spawning biomass required to produce MSY, and the relative fishing mortality, or the ratio of fishing mortality to the fishing mortality required to produce MSY. Estimation accuracies for 3 absolute quantities of interest, spawning biomass, recruitment, and fishing mortality, were also calculated for reference (Suppl. Materials). Overall, the performance measures indicate that the estimates of relative spawning biomass and relative fishing mortality had the level of accuracy expected for estimates of spawning potential and fishing intensity relative to MSY-based reference points used in analysis of domestic fisheries of the United States.

We used 6 simulation tests to evaluate the estimation performance of the MAS (Table 1). For these evaluations, we used 5 simulation test cases that come directly from Li et al. (2021), with our test cases 1–5 corresponding to their cases 1, 4, 7, 8, and 9 (Li et al., 2021, table 3), and our sixth case was based on a scenario for underreported catch. The test cases for examining the accuracy of estimates of stock status were as follows:

Case 1. The base case
Case 2. The case in which fishing mortality has a pattern like a roller coaster
Case 3. The case with fishing mortality set at a constant high level
Case 4. The case with dome-shaped fishery selectivity
Case 5. The case with an additional research survey
Case 6. The case in which true catch is underreported by 20%

Li et al. (2021) established the base test (case 1), which had one fishing fleet and one survey, with fully selected fishing mortality linearly increasing with time and a time-invariant logistic selectivity function for both the fishing fleet and the survey. Case 1 represents a baseline assessment scenario and the other tests are variants of case 1. Cases 2 and 3 were used to examine the effects of different fishing mortality patterns. In case 2, the fishing mortality pattern linearly increases to a high level and then declines to a low level. In case 3, the fishing mortality pattern is constantly at a high level. Cases 4 and 5 were used to examine how having dome-shaped fishery selectivity or an additional research survey index would affect estimation, respectively. With case 6, we examined the effects of underreported catch biomass by reducing the observed catch biomass to 80% of the true catch biomass simulated in the OM. Therefore, input data for cases 1–5 are unbiased, but case 6 includes biased catch data.

Results

Simulation tests were completed under the scenario for a population simulated on the basis of life history traits, with a value of 0.2 set for the standard deviation of log-scale recruitment, in the same OM used by Li et al. (2021). Results from those tests indicate that the MAS produced adequate estimation accuracy for stock status (Fig. 3). The magnitudes of estimation error were consistently low with MARE values for relative spawning biomass and relative fishing mortality on the order of 5% or less. Cases 1–5 are directly comparable to those in Li et al. (2021), which had similar estimation accuracies under the 4 assessment EMs they examined. The MRE values for cases 1–5 indicate that the estimates of relative spawning biomass and relative fishing mortality were unbiased with interquartile ranges of relative errors from about –5% to about 5% (Fig. 4), similar to what the MARE values for these cases indicate. Results for case 6, with biased catch data, indicate that the magnitude of estimation bias depended on the magnitude of the misreported catch biomass, which for this case was centered at 20% less than the true amount in the OM. The underreporting of catch led to underestimation of the absolute quantities of spawning biomass and recruitment by about 25% or roughly equal to the sum of the 20% underreporting

Table 1

Description of the 6 test cases used to evaluate the estimation accuracy of the Metapopulation Assessment System under the scenario of the operating model for a simulated population based on life history traits, with settings for recruitment variability (σ_R), deviations in fishing mortality (F), temporal patterns in F, selectivity patterns, number of surveys, survey coefficient of variation (CV), initial condition, and amount of underreported catch by test case. A dash denotes that the value or information is the same as that given for the base case (case 1). Note that $\emptyset_F$ denotes spawning biomass per recruit at F, $\emptyset_0$ denotes unfished spawning biomass per recruit, and F_{High} denotes F set at a high level.

Description	Case	σ_R	F deviations	F patterns	Selectivity patterns	Number of surveys	Survey CV	Initial condition	Misreported catch
Base case	1	0.2	Same per iteration	Increase[1]	Logistic	1	0.2	$\emptyset_F \neq \emptyset_0$	0%
Pattern in F	2	–	–	Roller coaster[2]	–	–	–	–	–
Time series	3	–	–	F_{High}	–	–	–	–	–
Domed selectivity	4	–	–	–	Double-logistic	–	–	–	–
Additional survey	5	–	–	–	–	2	–	–	–
Underreported catch	6	–	–	–	–	–	–	–	–20%

[1] F linearly increases from 0.01 to 0.40 over the assessment period.
[2] F linearly increases from 0.01 to F_{High} in year 24 and then linearly decreases to a low level (F_{Low}) in year 30, where F_{High} and F_{Low} produce 80% of maximum sustainable yield (MSY) with $F_{High} > F_{MSY}$ (the F that would result eventually in the MSY of a population) and $F_{Low} < F_{MSY}$, as in the yield curve depicted in figure 1 of Li et al. (2021).

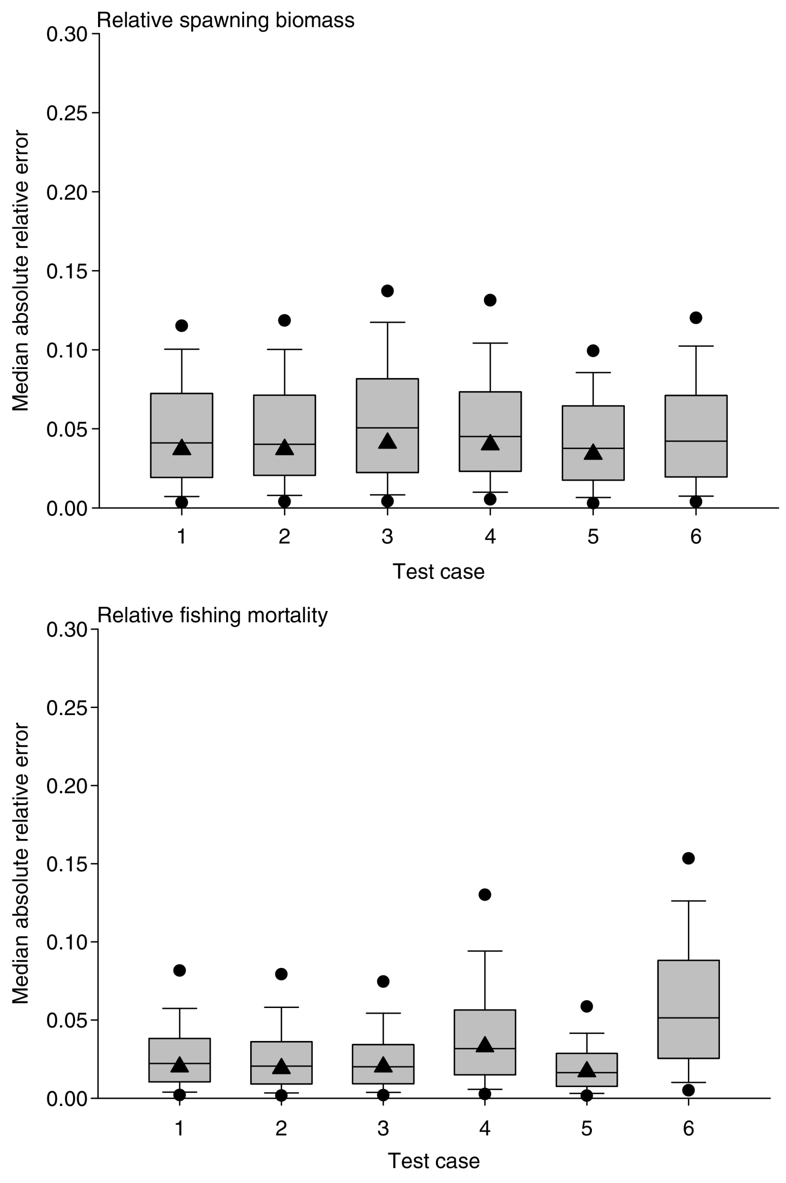

Figure 3

Box plots of the median absolute relative errors (MAREs) for estimation of relative spawning biomass, the ratio of spawning biomass to the spawning biomass required to produce maximum sustainable yield, and relative fishing mortality, the ratio of fishing mortality to the fishing mortality required to produce maximum sustainable yield, with 6 test cases in the Metapopulation Assessment System under the scenario for a simulated population based on life history traits. Case 1, the baseline assessment, involves one fishing fleet and one survey, with fully selected fishing mortality linearly increasing with time and a time-invariant logistic selectivity function for both the fishing fleet and the survey. In case 2, fishing mortality linearly increases to a high level and then declines to a low level. In case 3, fishing mortality is set at a constant high level. For case 4, fishery selectivity is dome shaped. Case 5 has an additional research survey. In case 6, true catch is underreported by 20%. In each box plot, the black line in the middle is the median. The upper and lower parts of each box represent the first and third quartiles (the 25th and 75th percentiles). The black circles indicate the 5th and 95th percentiles. The triangles indicate the minimum MARE values for tests 1–5 reported by Li et al. (2021). The whiskers extend above and below each box no more than 1.5 times the interquartile range.

and a baseline MARE of about 5% (Suppl. Materials, Suppl. Figs. 1 and 2). Overall estimation accuracy appeared adequate for stock status, and the results indicate that the performance of the MAS was similar to that of the 4 assessment platforms compared in Li et al. (2021), given the simulator assumptions and characteristics of the data collected.

Discussion

Lessons learned

We found that modular software infrastructure with good coding practices and model specification capacity increased the extensibility of a prototype for a next-generation stock assessment modeling system. Modularity entails having structured components for input-output processing, model specification and testing, and ensemble model capability. Extensibility means having components that can handle alternative spatiotemporal relations and interactions between populations, fleets, and environmental forcing, as well as having the capacity to reconfigure input data and include new observation types, such as close-kin mark-recapture data (Bravington et al., 2016). In this context, there is probably no single set of best practices to achieve modularity and extensibility, but we believe it is important to strive for the most advanced, yet acceptable software infrastructure to move forward. For the MAS, this goal led us to use GitHub, C++, the R language, and the Rcpp package to build the r4MAS package.

We have demonstrated that it is feasible to build a modeling platform that is based on R and C++ and is modular and fully extensible. For comparison, modularity is a structural feature of the design of AD Model Builder (ADMB) (Fournier et al., 2012), which is the modeling platform used in all the assessment models compared in Li et al. (2021). However, ADMB has limited extensibility because it does not support user-created C++ classes for extending the functionality of the modeling and estimation framework. The ADMB lacks this support because the platform is based on a template programming language that does not allow for additional OOP structuring in C++ (e.g., it does not allow inherited

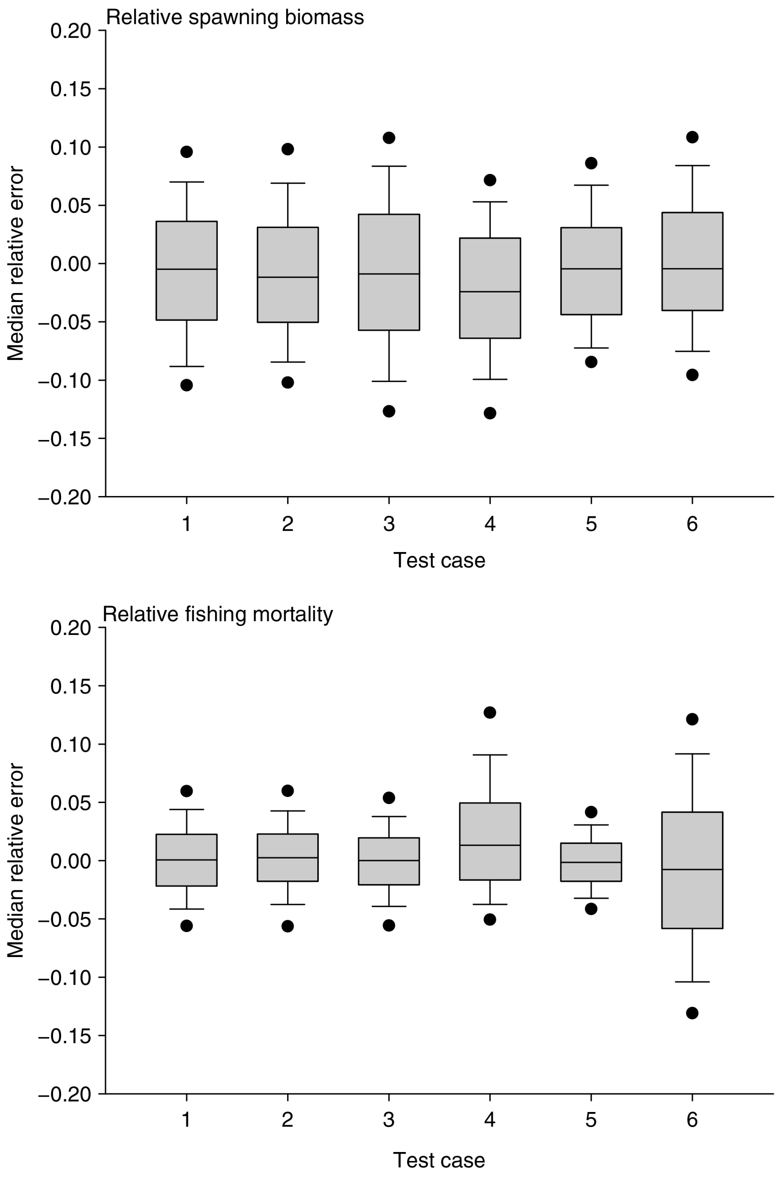

Figure 4

Box plots of the median relative errors (MREs) for estimation of relative spawning biomass, the ratio of spawning biomass to the spawning biomass required to produce maximum sustainable yield, and relative fishing mortality, the ratio of fishing mortality to the fishing mortality required to produce maximum sustainable yield, with 6 test cases in the Metapopulation Assessment System under the scenario for a simulated population based on life history traits. Case 1, the baseline assessment, involves one fishing fleet and one survey, with fully selected fishing mortality linearly increasing with time and a time-invariant logistic selectivity function for both the fishing fleet and the survey. In case 2, fishing mortality linearly increases to a high level and then declines to a low level. In case 3, fishing mortality is set at a constant high level. For case 4, fishery selectivity is dome-shaped. Case 5 has an additional research survey. In case 6, true catch is underreported by 20%. In each box plot, the black line in the middle is the median. The upper and lower parts of each box represent the first and third quartiles (the 25th and 75th percentiles). The black circles indicate the 5th and 95th percentiles. The whiskers extend above and below each box no more than 1.5 times the interquartile range.

or polymorphic objects). Further, ADMB development ended as of March 2024. In contrast, the MAS depends on the ATL, which fully supports C++ extensibility, is not limited by a template programming language requirement, and can be modified to use alternative class structures or optimizers, including neural network optimization.

Systematic testing should cover the range of possible assessment modeling situations, including models that include biased data (i.e., case 6). Testing expected performance of an EM when input data are imprecise or biased is important for characterizing the trade-offs of alternative EM structures. Under case 6, use of the underreported catch data led to consistent underestimation of absolute quantities of spawning biomass and recruitment (Suppl. Materials, Suppl. Figs. 1 and 2) but resulted in reasonably accurate estimates of the scaled quantities of fishing mortality, relative spawning biomass, and relative fishing mortality (Figs. 3 and 4). Overall, simulation tests should strive to evaluate the predictive accuracy over a wide range of expected assessment conditions.

Many of the features and technical requirements for a next-generation system were elucidated in Punt et al. (2020). These insights provide solid technical guidance, but we believe software engineering will also play a key role in building a successful next-generation system. This is because extensibility to incorporate new data or information sources and to adapt to new analytical techniques, or knowledge acquisition models, will be essential for optimizing future assessment system performance. In this context, the r4MAS package makes model setup intuitive and is easy to use, an important user-oriented feature for any next-generation modeling platform.

The MAS project has illustrated the need to adapt the software infrastructure to the expected modeling needs and future requirements. The MAS prototype was designed to employ a mixture of both functional programming and OOP paradigms to produce a modular and extensible software system composed of code in the C++, Rcpp, and R languages. Although the MAS project lacked resources to develop an extensive input-output processing system, such as the R package r4ss (Taylor et al., 2021; available from website), this functionality would be an ongoing requirement that would be expected to be addressed in the system life cycle. The focus on future extensibility of the MAS aligns with the general point that having a development team with diverse backgrounds and skill sets is important for enhancing creativity, flexibility, and productivity in new product management. In this context, software engineering is a key technical requirement to augment the features emphasized in the most recent workshop on next-generation assessment models (e.g., Hoyle et al.[1]; Punt et al., 2020). Other important software engineering components include object-oriented design, software unit testing requirements, parallel processing capacity, R language interfacing for input and output processing, and long-term extensibility with C++ template metaprogramming, as well as the capacity to efficiently handle ensemble model analyses, apply model averaging, and conduct management strategy evaluations. These engineering components will improve the practical implementation of any next-generation system.

Another lesson learned is that both constancy of purpose and adequate resources will be needed to efficiently produce a next-generation stock assessment modeling platform. This point is evident for the successful and ongoing Stock Synthesis project (Methot and Wetzel, 2013; available from website), which has had adequate resources and funding for student and postdoctoral contributors for many years. In comparison, the MAS project was allocated limited resources and was also affected by staff turnover, loss of institutional knowledge, and competing time commitments. When these limitations were recognized, the NOAA Stock Assessment Improvement Plan was updated to focus on a next-generation assessment system (Lynch et al., 2018). This emphasis on a next-generation platform led to the creation of the Fisheries Integrated Modeling System (FIMS) (FIMS Implementation Team, 2022), the development of which was initiated in 2020. Since then, the MAS prototype has been a research and development pipeline to the FIMS. Resources for the MAS have transitioned to focus solely on FIMS implementation. The FIMS project is developing the next-generation stock assessment platform supported by the National Marine Fisheries Service. To achieve this end, DevOps workflow approaches, such as GitHub, are actively used for the FIMS project—to efficiently build a next-generation assessment system that is agile and responsive to diverse needs of users at the National Marine Fisheries Service and other fisheries agencies. In this context, we emphasize that allocating sufficient resources and maintaining a constancy of purpose will be essential for the ongoing success of the FIMS project.

The technical influence of the MAS on the FIMS is significant in several aspects. Both systems share a foundation in object-oriented inheritance and similar design patterns, facilitating extensible module development and ensuring ease of unit testing. These systems have broad applicability and long-term viability because the principles of portability, maintainability, and extensibility are emphasized. Neither system is tied to a single estimation engine, enhancing their flexibility. Additionally, both systems can be compiled with the ATL and TMB, underlining their robust structural design as modeling frameworks rather than as static models. The interface of the Rcpp package enables seamless integration with R, with the FIMS project adopting a refactored naming convention to maintain an essentially identical interface to the MAS. This interface bridges R with underlying C++ code and reinforces the systems' interoperability and user accessibility.

Both the FIMS and MAS are designed to support complex metapopulation modeling, which includes handling multiple populations, fleets, surveys, and areas and sex differentiation. This capacity is essential for comprehensive fisheries management and assessment. Both systems can execute model runs in parallel, significantly enhancing computational efficiency. In terms of development cycles, the MAS is characterized by a highly agile development approach. The FIMS follows a similar agile method, allowing iterative improvements and rapid adaptation to new requirements. The shared technical characteristics and development philosophies between the MAS and FIMS highlight the substantial influence the MAS has exerted on the design and functionality of the FIMS, making the FIMS a versatile and extensible tool for fisheries modeling and assessment.

Future challenges

In the future, Breiman's (2001) 2 cultures of data modeling and machine learning modeling may need to be combined to produce the best predictive accuracy for integrated stock assessment models. This approach would contrast with the long-term fisheries modeling practice of deploying only statistical models to characterize the likelihood of observed data in stock assessment applications. The number of potential applications of machine learning models to directly support and inform fisheries stock assessments is increasing as automated data collection and the sizes of data sets on marine communities and environments increase (Malde et al., 2020). More frequent applications of machine learning for stock assessments can be expected. Such models have produced accurate predictions without explicit structure for species identification, discard estimation, and characterization of size composition (Beyan and Browman, 2020). Recent applications of machine learning for components of a next-generation assessment system include forecasting recruitment—an essential component of future stock projections (Smoliński, 2019); estimating natural mortality rates—a key life history parameter for age-structured stock assessments (Liu et al., 2020); and setting relative biomass prior distributions for the catch-only CMSY++ assessment model (Froese et al., 2023). Although there is no certainty about the future needs for stock assessment modeling, an important future challenge is to ensure that any next-generation assessment platform can be extended to incorporate data and assessment model inputs produced by machine learning algorithms in an appropriate integrated estimation framework.

In recent years, it has become evident that the capacity to do ensemble modeling is likely important for structuring a next-generation assessment system (e.g., Lynch et al., 2018). Providing this capacity represents a major challenge given the multiplicity of assessment model structures that are consistent with typical observed data and knowledge for a fishery system. Ensemble model and multi-model inference are practical tools for implementing ecosystem-based fishery management in the face of uncertain system dynamics. An ongoing challenge will be to represent the fishery system with parsimonious but representative structures that address the trade-offs between the bias and variance of individual model results and the covariances between model results when combined to make ensemble model assessments and projections.

Some of the challenges we encountered in building a prototype of a next-generation modeling system naturally led to some unresolved issues. How to produce an analytical system whose components are maintainable and extensible and yet as low-cost and efficient as possible? To address this issue, the C++ language was used for coding the calculation engine and the R language was used for coding the system interface. How to incorporate diverse approaches for assessment model structure and stock status determination conditioned on available information on fishery systems ranging from data limited to data rich? We addressed this challenge by using template metaprogramming to produce a modular and extensible assessment system. How to ensure that diverse model structures and outputs are reproducible, well-documented, and transparent to support reliable scientific computing and open science practices for stock assessments (Lowndes et al., 2017; Wilson et al., 2017; Ramachandran et al., 2021). To address this issue, we used GitHub to provide version control and to archive source code and analytical approaches. These challenges underscore the point that any engineering strategy for building a next-generation system needs to be adaptive and will continually evolve.

Conclusions

In summary, our proposed engineering strategy includes specifications for the software life cycle, engineering, and infrastructure design needed to support a spatially explicit, next-generation stock assessment system, the MAS. Software infrastructure was designed by using OOP principles of encapsulation, data abstraction, polymorphism, and inheritance. Three primary components were used to implement the software infrastructure: GitHub for collaboration, version control, and code organization; the C++ language for coding fishery system dynamics and estimation modeling; and the R language for coding the input-output interface and systematic testing. Systematic testing was a key strategic element for iterative improvement under the life cycle of this next-generation stock assessment modeling platform.

Resumen

Los procesos ambientales y antropogénicos han provocado cambios generalizados en la productividad y la distribución espacial de los recursos pesqueros marinos. A medida que cambien las distribuciones geográficas de las poblaciones de peces y, por consiguiente, de las flotas pesqueras, será necesario mejorar los datos espaciotemporales y la modelación espacial para estimar la abundancia y la productividad. El objetivo de este artículo es proponer una estrategia para diseñar una plataforma de modelación espacialmente explícita para modelos de nueva generación para evaluación de poblaciones. Presentamos nuestro enfoque para desarrollar un prototipo de sistema, el Sistema de Evaluación de Metapoblaciones (MAS), que es fácil de usar, modular y extensible. El prototipo MAS se diseñó para dar soporte a modelos metapoblacionales complejos, que incluyen el manejo de múltiples poblaciones, áreas, flotas, prospecciones, así como la diferenciación de sexos. Describimos los componentes del ciclo de vida del software, la ingeniería y el diseño de la infraestructura para dar soporte a un sistema de nueva generación para la evaluación espacialmente explícita de poblaciones. La infraestructura de software se diseñó e implementó con 3 componentes: GitHub para la colaboración, el control de versiones y la organización del código; el lenguaje C ++ para codificar la dinámica del sistema pesquero y modelado de estimadores; y el lenguaje R para codificar la interfaz de entrada-salida y las pruebas sistemáticas. Las pruebas sistemáticas fueron un componente clave del ciclo de vida de desarrollo del MAS y se aplicaron para garantizar que los requisitos del sistema, como la estimación precisa de las cantidades de interés, se implementaran con éxito. Concluimos con la discusión de algunas lecciones aprendidas y futuros retos para los esfuerzos en curso para implementar una plataforma de evaluación de poblaciones de nueva generación.

Acknowledgments

We thank C. Porch, K. Shertzer, P. Lynch, and R. Methot for their insights and helpful discussions on the MAS project. For this research, no specific grants were received from funding agencies in the public, commercial, or not-for-profit sectors.

Literature cited

Auger-Méthé, M., K. Newman, D. Cole, F. Empacher, R. Gryba, A. A. King, V. Leos-Barajas, J. M. Flemming, A. Nielsen, G. Petris, et al.
2021. A guide to state–space modeling of ecological time series. Ecol. Monogr. 91(4):e01470. Crossref

Berger, A. M., D. R. Goethel, and P. D. Lynch.
2017 Introduction to "Space oddity: recent advances incorporating spatial processes in the fishery stock assessment and management interface." Can. J. Fish. Aquat. Sci. 74:1693–1697. Crossref

Beyan, C., and H. I. Browman.
2020. Setting the stage for the machine intelligence era in marine science. ICES J. Mar. Sci. 77:1267–1273. Crossref

Bravington, M. V., H. J. Skaug, and E. C. Anderson.
2016. Close-kin mark-recapture. Stat. Sci. 31:259–274. Crossref

Breiman, L.
2001. Statistical modeling: the two cultures. Stat. Sci. 16:199–231. Crossref

Brodziak, J., and J. Link.
2002. Ecosystem-based fishery management: what is it and how can we do it? Bull. Mar. Sci. 70:589–611.

Brodziak, J., and K. Piner.
2010. Model averaging and probable status of North Pacific striped marlin, *Tetrapturus audax*. Can. J. Fish. Aquat. Sci. 67:793–805. Crossref

Brodziak, J., M. Supernaw, and C. Porch.
2014. Object-oriented design of MAS, a metapopulation assessment system. 144th American Fisheries Society Annual Meeting; Quebec City, 17–21 August. Am. Fish. Soc., Bethesda, MD. [Abstract.] [Available from website.]

Brodziak, J., T. A'mar, and M. Supernaw.
2017. Designing a general MSE framework for a movement-based metapopulation assessment system. 147th American Fisheries Society Annual Meeting; Tampa, 18–24 August. Am. Fish. Soc., Bethesda, MD. [Abstract.] [Available from website.]

Brooks, E. N., and J. K. T. Brodziak.
2024. Simulation testing performance of ensemble models when catch data are underreported. ICES J. Mar. Sci. 81:1053–1072. Crossref

Burnham, K. P., and D. R. Anderson.
2002. Model selection and multimodel inference: a practical information-theoretic approach, 2nd ed., 488 p. Springer, New York.

Carruthers, T. R., and A. R. Hordyk.
2018. The data-limited methods toolkit (DLMtool): an R package for informing management of data-limited populations. Methods Ecol. Evol. 9:2388–2395. Crossref

Cheung, W. W. L., R. Watson, and D. Pauly.
2013. Signature of ocean warming in global fisheries catch. Nature 497:365–368. Crossref

Demidenko, E.
2004. Mixed models: theory and application, 704 p. John Wiley & Sons, Hoboken, NJ.

Deming, W. E.
1982. Out of the crisis, 507 p. MIT Press, Cambridge, MA.

Doonan, I., K. Large, A. Dunn, S. Rasmussen, C. Marsh, and S. Mormede.
2016. Casal2: New Zealand's integrated population modelling tool. Fish. Res. 183:498–505. Crossref

Dormann, C. F., J. M. Calabrese, G. Guillera-Arroita, E. Matechou, V. Bahn, K. Bartoń, C. M. Beale, S. Ciuti, J. Elith, K. Gerstner, et al.
2018. Model averaging in ecology: a review of Bayesian, information-theoretic, and tactical approaches for predictive inference. Ecol. Monogr. 88:485–504. Crossref

Ducharme-Barth, N. D., and M. T. Vincent.
2022. Focusing on the front end: a framework for incorporating uncertainty in biological parameters in model ensembles of integrated stock assessments. Fish. Res. 255:106452. Crossref

Eddelbuettel, D.
2013. Seamless R and C++ integration with Rcpp, 220 p. Springer, New York.

Eubank, R. L, and A. Kupresanin.
2011. Statistical computing in C++ and R, 540 p. CRC Press, Boca Raton, FL.

FIMS Implementation Team.
2022. NOAA-FIMS: the NOAA Fisheries Integrated Modeling System. R package, vers. 0.0.0.9000. [Available from website.]

Fournier, D., and C. P. Archibald.
1982. A general theory for analyzing catch at age data. Can. J. Fish. Aquat. Sci. 39:1195–1207. Crossref

Fournier, D. A., H. J. Skaug, J. Ancheta, J. Ianelli, A. Magnusson, M. N. Maunder, A. Nielsen, and J. Sibert.
2012. AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex non-linear models. Optim. Methods Softw. 27:233–249. Crossref

Free, C. M., J. T. Thorson, M. L. Pinsky, K. L. Oken, J. Wiedenmann, and O. P. Jensen.
2019. Impacts of historical warming on marine fisheries production. Science 363:979–983. Crossref

Froese, R., H. Winker, G. Coro, M.-L. D. Palomares, A. C. Tsikliras, D. Dimarchopoulou, K. Touloumis, N. Demirel, G. M. S. Vianna, G. Scarcella, et al.
2023. New developments in the analysis of catch time series as the basis for fish stock assessments: the CMSY++ method. Acta Ichthyol. Piscat. 53:173–189. Crossref

Griewank, A., and A. Walther.
2008. Evaluating derivatives: principles and techniques of algorithmic differentiation, 2nd ed., 438 p. Soc. Ind. Appl. Math., Philadelphia, PA.

Gullestad, P., S. Sundby, and O. S. Kjesbu.
2020. Management of transboundary and straddling fish stocks in the Northeast Atlantic in view of climate-induced shifts in spatial distribution. Fish Fish. 21:1008–1026. Crossref

Jana, D.
2005. C++ and object-oriented programming paradigm, 2nd ed., 531 p. Raj Press, New Delhi, India.

Jardim, E., M. Azevedo, J. Brodziak, E. N. Brooks, K. F. Johnson, N. Klibansky, C. P. Millar, C. Minto, I. Mosqueira, R. D. M. Nash, et al.
2021. Operationalizing ensemble models for scientific advice to fisheries management. ICES J. Mar Sci. 78:1209–1216. Crossref

Kell, L. T., I. Mosqueira, P. Grosjean, J.-M. Fromentin, D. Garcia, R. Hillary, E. Jardim, S. Mardle, M. A. Pastoors, J. J. Poos, et al.
2007. FLR: an open-source framework for the evaluation and development of management strategies. ICES J. Mar. Sci. 64:640–646. Crossref

Koenigstein, S., F. C. Mark, S. Gößling-Reisemann, H. Reuter, and H.-O. Poertner.
2016. Modelling climate change impacts on marine fish populations: process-based integration of ocean warming, acidification and other environmental drivers. Fish Fish. 17:972–1004. Crossref

Kristensen, K., A. Nielsen, C. W. Berg, H. Skaug, and B. M. Bell.
2016. TMB: automatic differentiation and Laplace approximation. J. Stat. Softw. 70(5):1–21. Crossref

Laplante, P. A. (ed.).
2001. Comprehensive dictionary of computer science, engineering and technology, 560 p. CRC Press, Boca Raton, FL.

Legault, C. M., and V. R. Restrepo.
1998. A flexible forward age-structured assessment program. Collect. Vol. Sci. Pap. ICCAT 49:246–253.

Li, B., K. W. Shertzer, P. D. Lynch, J. N. Ianelli, C. M. Legault, E. H. Williams, R. D. Methot Jr., E. N. Brooks, J. J. Deroba, A. M. Berger, et al.
2021. A comparison of 4 primary age-structured stock assessment models used in the United States. Fish. Bull. 119:149–167. Crossref

Liu, C., S. Zhou, Y.-G. Wang, and Z. Hu.
2020. Natural mortality estimation using tree-based ensemble learning models. ICES J. Mar. Sci. 77:1414–1426. Crossref

Lowndes, J. S. S., B. D. Best, C. Scarborough, J. C. Afflerbach, M. R. Frazier, C. C. O'Hara, N. Jiang, and B. S. Halpern.
2017. Our path to better science in less time using open data science tools. Nat. Ecol. Evol. 1:0160. Crossref

Lynch, P. D., R. D. Methot, and J. S. Link (eds.).
2018. Implementing a next generation stock assessment enterprise. An update to the NOAA Fisheries Stock Assessment Improvement Plan. NOAA Tech. Memo. NMFS-F/SPO-183, 127 p.

Malde, K., N. O. Handegard, L. Eikvil, and A.-B. Salberg.
2020. Machine intelligence and the data-driven future of marine science. ICES J. Mar. Sci. 77:1274–1285. Crossref

Martin, R. C.
2009. Clean code: a handbook of agile software craftsmanship, 431 p. Pearson Educ., Boston, MA.

Maureaud, A. A., R. Frelat, L. Pécuchet, N. Shackell, B. Mérigot, M. L. Pinksy, K. Amador, S. C. Anderson, A. Arkhipkin, A. Auber, et al.
2021. Are we ready to track climate-driven shifts in marine species across international boundaries? - A global survey of scientific bottom trawl data. Global Change Biol. 27:220–236. Crossref

Methot, R. D.
1990. Synthesis model: an adaptable framework for analysis of diverse stock assessment data. Int. North Pac. Fish. Comm. Bull. 50:259–277.

Methot, R. D., Jr., and C. R. Wetzel.
2013. Stock synthesis: a biological and statistical framework for fish stock assessment and fishery management. Fish. Res. 142:86–99. Crossref

Meyer, R., and R. B. Millar.
1999. BUGS in Bayesian stock assessments. Can. J. Fish. Aquat. Sci. 56:1078–1087. Crossref

Miller, T. J., J. A. Hare, and L. A. Alade.
2016. A state-space approach to incorporating environmental effects on recruitment in an age-structured assessment model with an application to southern New England yellowtail flounder. Can. J. Fish. Aquat. Sci. 73:1261–1270. Crossref

National Research Council.
2012. Assessing the reliability of complex models: mathematical and statistical foundations of verification, validation, and uncertainty quantification, 106 p. Natl. Acad. Press, Washington, DC. [Available from website.]

Nielsen, A., and C. W. Berg.
2014. Estimation of time-varying selectivity in stock assessments using state-space models. Fish. Res. 158:96–101. Crossref

Palacios-Abrantes, J., S. Crosson, C. Dumas, R. Fujita, A. Levine, C. Longo, and O. P. Jensen.
2023. Quantifying fish range shifts across poorly defined management boundaries. PLoS ONE 18(1):e0279025. Crossref

Patterson, K., R. Cook, C. Darby, S. Gavaris, L. Kell, P. Lewy, B. Mesnil, A. Punt, V. Restrepo, D. W. Skagen, and G. Stefánsson.
2001. Estimating uncertainty in fish stock assessment and forecasting. Fish Fish. 2:125–157. Crossref

Perry, A. L., P. J. Low, J. R. Ellis, and J. D. Reynolds.
2005. Climate change and distribution shifts in marine fishes. Science 308:1912–1915. Crossref

Peterman, R. M.
2004. Possible solutions to some challenges facing fisheries scientists and managers. ICES J. Mar. Sci. 61:1331–1343. Crossref

Pinsky, M. L., R. L. Selden, and Z. J. Kitchel.
2020. Climate-driven shifts in marine species ranges: scaling from organisms to communities. Annu. Rev. Mar. Sci. 12:153–179. Crossref

Privitera-Johnson, K. M., and A. E. Punt.
2020. A review of approaches to quantifying uncertainty in fisheries stock assessments. Fish. Res. 226:105503. Crossref

Punt, A. E.
2019. Spatial stock assessment methods: a viewpoint on current issues and assumptions. Fish. Res. 213:132–143. Crossref

Punt, A. E., and R. Hilborn.
1997. Fisheries stock assessment and decision analysis: the Bayesian approach. Rev. Fish Biol. Fish. 7:35–63. Crossref

Punt, A. E., A. Dunn, B. Elvarsson, J. Hampton, S. D. Hoyle, M. N. Maunder, R. D. Methot, and A. Nielsen.
2020. Essential features of the next-generation integrated fisheries stock assessment package: a perspective. Fish. Res. 229:105617. Crossref

R Core Team.
2021. R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. [Available from website, accessed November 2021].

Ramachandran, R., K. Bugbee, and K. Murphy.
2021. From open data to open science. Earth Space Sci. 8:e2020EA001562. Crossref

Shin, Y.-J., and P. Cury.
2004. Using an individual-based model of fish assemblages to study the response of size spectra to changes in fishing. Can. J. Fish. Aquat. Sci. 61:414–431. Crossref

Smith, S. J., J. J. Hunt, and D. Rivard (eds.).
1993. Risk evaluation and biological reference points for fisheries management. Can. Spec. Publ. Fish. Aquat. Sci. 120, 436 p.

Smoliński, S.
2019. Incorporation of optimal environmental signals in the prediction of fish recruitment using random forest algorithms. Can. J. Fish. Aquat. Sci. 76:15–27. Crossref

Stroustrup, B.
2013. The C++ programming language, 4th ed., 1346 p. Addison-Wesley, Boston, MA.

Supernaw, M., C. Stawitz, and B. Li.
2022. r4MAS: read and write input and output for MAS model. R package. [Available from website.]

Taivalsaari, A.
1996. On the notion of inheritance. ACM Comput. Surveys 28:438–479. Crossref

Taylor, I. G., K. L. Doering, K. F. Johnson, C. R. Wetzel, and I. J. Stewart.
2021. Beyond visualizing catch-at-age models: lessons learned from the r4ss package about software to support stock assessments. Fish. Res. 239:105924. Crossref.

Waterhouse, L., D. B. Sampson, M. Maunder, and B. X. Semmens.
2014. Using areas-as-fleets selectivity to model spatial fishing: asymptotic curves are unlikely under equilibrium conditions. Fish. Res. 158:15–25. Crossref

Wickham, H.
2011. testthat: get started with testing. R J. 3(2):5–10. Crossref
2015. R packages: organize, test, document, and share your code, 182 p. O'Reilly Media, Sebastopol, CA.
2016. ggplot2: elegant graphics for data analysis, 2nd ed., 260 p. Springer, New York.

Wickham, H., and G. Grolemund.
2017. R for data science: import, tidy, transform, visualize, and model data, 492 p. O'Reilly Media, Sebastopol, CA.

Wilson, G., D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, et al.
2014. Best practices for scientific computing. PLoS Biol. 12(1):e1001745. Crossref

Wilson, G., J. Bryan, K. Cranston, J. Kitzes, L. Nederbragt, and T. K. Teal.
2017. Good enough practices in scientific computing. PLoS Comput. Biol. 13(6):e1005510. Crossref

Xie, Y.
2015. Dynamic documents with R and knitr, 2nd ed., 294 p. CRC Press, Boca Rotan, FL.

Xie, Y., J. J. Allaire, and G. Grolemund.
2019. R markdown: the definitive guide, 303 p. CRC Press, Boca Rotan, FL.

Vandevoorde, D., N. M. Josuttis, and D. Gregor.
2018. C++ templates: the complete guide, 2nd ed., 788 p. Addison-Wesley, New York.