Focus Questions

CAPAM workshop on the creation of frameworks for the next generation general stock assessment models

Coding philosophies and software structure

Should the next generation model be coded by stock assessment scientists or professionally trained computer programmers?

Currently, most general stock assessment packages have been coded by stock assessment scientists, but some have been coded by professional computer programmers. The stock assessment scientists' backgrounds in programming have varied from self-taught to computer science degrees. However, code written by stock assessment scientists is often poorly written and documented, by professional software development standards. For a collaborative project to be successful, code needs to be well organized and documented. This either means using professional programmers or extensively training stock assessment scientists.

On the one hand, it is uncertain whether training of stock assessment scientists will be successful or cost effective. On the other hand, professional programmers who do not have stock assessment expertise would require comprehensive specification documents and testing procedures to ensure that the model is implemented correctly. Also, collaboration will be less direct since other contributors will also be limited to writing specification documents rather than hands-on coding. Another issue is that creativity and new ideas are often a consequence of a stock assessment scientist implementing and testing their own model, which will be lost if professional programmers are used. Nevertheless, it can be argued that the next generation stock assessment model should first be an efficient, flexible and reliable tool for assessing stocks and providing management advice. This is more important than its role as a platform for developing new approaches.

There is a variety of opinions on this topic. Some think that the clear answer is both. It will take a village to build the nextgen model well. On the other-hand, others think that training a stock assessment scientist is unlikely to be successful or cost effective. For example, attempts to develop complex Casal2 functionality by people with limited training caused execution time to increase by 8x.

Going further up than a professional programmer you need to also consider Software Architecture as a discipline. The way you design the overall structure of your system before you start writing any code is going to have a significant impact on your quality.

Casal2 is a system for making models. It can be thought of as a system that compiles and runs a model based on "code" entered through the configuration file. A computer scientist wanting to add a new process to Casal2 only needs to add 2 new files and add 2 lines of code to a third. All linking of their code through the system, including configuration file and error handling, is handled by the Casal2 platform.

Casal2 has been designed to separate the framework code from the modeling code. Casal2 provides a harness for scientists to contribute code in specific (easily accessed) locations with a huge amount of framework automatically provided.

Systems like Casal2 can be built to support the running of external scripts for computer scientists to easily write and verify code in a non-performant manner before having it written and optimized by a programmer. A computer scientist could write a simple python script that would act as a new custom process in Casal2 with no Casal2 code changes required (not yet implemented). Once the computer scientist is happy with their python script, they can hand it off to a programmer with test cases for it to be written in C++ and optimized.

Casal2 is designed to allow scientists to do science first, but this requires programmers to do programming.

There's an important distinction between the science of finding the best model among a well-understood set of options for a production assessment application used in management and the science of exploring new modeling approaches that may have benefit for future applications. The first case clearly benefits from a generalized model that is efficient and well tested. The latter benefits from the scientist having the ability to directly change the model in fundamental ways that are deeper than adding a new functional form for some process. It's not necessary to use the same model for both types of science, but those exploring new approaches typically need to have a solid grounding in production assessments for their research to be grounded in reality, and there are inefficiencies associated with having a separate platform for productions assessments and exploratory research.

One approach, which may not work well, is hiring professional programmers for a short-term contracts without having professional expertise available on an ongoing basis. These contracts could be via the original developers or others that have adequate understanding to change and improve the software

How can we ensure that with all the desired features included the code remain computationally efficient for models that don't use the more computationally demanding features?

The more complex the features that are included, the more computationally intensive the calculations will become. For example, including random effects is notorious for making models impractically computationally demanding and causing convergence issues. Therefore, algorithms and code need to be as computationally efficient as possible. For some cases computational approximations may be needed, whereas the exact calculations may also be needed for simpler models or for evaluating the size of approximation errors. Alternatively, the addition of a feature may cause calculations to become more demanding even when the feature is not used, due to changes to the base code used for all models. In such cases redundant code may be needed such that one version of the code is applied if the feature is used, and another version if the feature is not used.

Professional programming is probably the pathway to achieving efficiency even when complex options are available. Computational efficiency is not the only relevant metric here. Experience with existing

general models is that high diversity of options opens possibilities for users to select inappropriate combinations, or worse, for some combinations to create incorrect answers.

The architecture of your modeling system is very important here. A separation of the code that does the computations (science) from the code that builds the model is key. Casal2 compiles and runs a model based on the configuration file. No code that is not part of the model is used. All of the processes in Casal2 has been written as a state-machine to ensure the least amount of code is executed at any given time.

It might be worth considering code that compiles itself such that only the features that are desired are included in the model and then the code is compiled. This also may allow for additional features to be added efficiently at compile/runtime.

Is there a coding standard that is appropriate for the development of the next generation model?

Most general stock assessment models have been developed without the use of a rigorous coding standard. Successful development of the next generation stock assessment model using a collaborative approach requires adoption of a coding standard to facilitate contributions by multiple coders as well as for continued maintenance and development. Coding standards are well developed in the field of computer science, and appropriate standards should be adopted.

Coding standards (how the code should be written) are a Programming 101 paradigm. Emphasis should be given to how you structure unit tests (e.g. to use a mocking framework, configuration files, integration tests, full test models etc). Casal2 use the Google C++ coding style

Is there a way to easily allow the addition of new features?

The addition of new features to a general model generally requires adding code to the model and this greatly limits the number of people who can contribute. To facilitate the ability of assessment authors to add features, alternative approaches should be considered. For example, there are a variety of selectivity, growth, and stock recruitment curves that assessment authors might consider, and they may wish to test newly developed curves. If the general model had the capability to read in pseudocode that described the desired curve and to then interpret it into the underlying programming language of the general model (with or without recompilation), this would allow assessment authors to easily make simple changes to the general model. A simple example approach would be to allow the input of R-formatted GLM type code to link parameters to covariates.

Casal2 uses standard software architecture patterns. Adding a new object to Casal2 requires the addition of 2 lines of code into the relevant “factory” class. You can literally add a new nop (do nothing) process to Casal2 in 2 minutes, then focus on writing the actual scientific logic.

Underlying language base

What language should be used for the basis of the general model?

General stock assessment models require efficient calculations to ensure that the complex models can be implemented. In general, the calculations involve optimizing an objective function (e.g. the joint likelihood of the fit to the data) to estimate the model parameters. However, other options are also implemented such as Bayesian integration. ADMB has been the dominant language underlying many general models due to its use of automatic differentiation, which is efficient for optimization. However, the recent trend towards state-space models (inclusion of random effects to represent process variation) has caused a move towards the use of TMB. Other versions of similar underlying programming languages have been developed and used for general stock assessment models (e.g. CASAL). ADMB and TMB are high level programming languages and based on other more standard programming languages (e.g. C++).

It is not clear if the next generation general model should be based on one of the existing programming languages used for stock assessment models (e.g. ADMB or TMB), an existing language not currently used for stock assessment, or a new language developed. There is potential for the next generation model to be highly computationally intensive in some applications, particularly if random effects are used. Therefore, further development of the underlying programing language may be required.

With consumer CPUs having 24+ cores now, there is a significant shift in paradigm for how software, especially high performance, is written. The use of multi-core CPUs and GPUs has provided a method of speeding calculations up by orders of magnitude. Automatic differentiation is typically a linear method and with newer CPUs the speed increase by using it is now lost when compared to multi-threaded algorithms.

New modeling systems should be multi-language. Your core engine should be written in a way that allows key people to optimize it against profiling (something like C++ is ideal here because it's so close to the CPU). Ease-of-use mechanics should be provided through interfaces with other languages (e.g. Lua, Python, Ruby etc) where execution can be handed off to slower code that is easier to write and iterate.

It's also worth considering the ability to find people with significant expertise in the languages being used. Python developers are very common.

Hosting the project

What kind of code testing will ensure that the model has been implemented correctly?

Any next generation general model will require intensive testing and validation to ensure that the models are implemented correctly. Errors could be in both the specification documentation (if professional programmers are used) and in the actual code. Testing stock assessment models is complicated when results are not known in advance. This differs from testing where the errors are easy to see in the user interface, functionality, or well-established examples where calculations are simply repeated with different quantities. One approach that has been used is dual implementation in two completely different approaches (e.g. ADMB and Excel) where the same coding errors in both

approaches is less likely. This works well for simple models, or subcomponents of moderately complex models, but may be difficult in a comprehensive next generation model.

One approach is to use case studies from existing stock assessment packages to test the next generation model. This approach requires exactly the same specifications among programs, with the same assumptions or approximations, so it requires the packages to be very similar. Also, new developments could not be tested this way and would require independent coding to test them. Another approach would be completely independent implementation by parallel development teams in the same programming language, which is common in commercial software development. However, this approach may be prohibitively expensive.

In addition to testing new features, old features need to be checked to make sure the new code does not break the old code. Therefore, there should be automatic testing (unit tests) using a set of comprehensive examples that cover the complete feature range to test against known results.

Casal2 has extensive unit tests covering the core platform. The core platform has significant code coverage (i.e. coverage of every line of code) with tests for many scenarios. Every build of Casal2 runs these unit tests to determine if the build is a success of failure. The scale of unit testing starts at testing a single line or equation with multiple knowing inputs and outputs all the way up to running multiple models through different run modes with expected outcomes.

Casal2 developers are also looking at putting in “stub-models”, specific models where the outcome isn’t 100% known but can be approximated (e.g. running MCMC to approximate Pi). Algorithms can be run against this to determine a measure of confidence.

New algorithms can be implemented and tests with knowing good inputs and outputs are used. Every single equation can be checked in isolation to ensure it’s consistent, increasing the chance that the sum of all pieces is equally consistent.

Stock assessment model features

Do surplus production models and delay-difference model need to be explicitly included or is an age-structured model adequate?

Surplus production models and delay difference models are still used for stock assessment of several stocks. These approaches have the advantage that they do not need an understanding of the population dynamics subprocesses nor do they need catch composition data. In addition, it has been argued that these simple models can perform better in a management context than more complex models. Surplus production models can be approximated by an age structured model (although the age dynamics make the production curve change over time) and delay difference models can be represented exactly using an age-structured model (I think). Therefore, is it necessary to also explicitly include production models and delay-difference models in a general stock assessment package? A separate package for these types of models could be used, but this would make comparisons between the simple models and the age-

structured models less convenient because the outputs and diagnostics available in the general model would have to be developed for the independent package.

Does the general model need to include a VPA type approach?

Virtual Population Analysis (VPA) or cohort analysis is a common age-structured modelling approach to analysis of catch-at-age data. One main feature of VPA is that fishing mortality varies by age and year and catch (at age) is combined across all fisheries despite possible large differences in their selectivities. In contrast, statistical integrated stock assessment models separate catch into fisheries that have different selectivities and force structure on the selectivity (e.g. time invariant functional forms). VPAs can be approximated in integrated analysis by allowing temporal variation in the selectivity. At the extreme an independent parameter can be estimated for each age and year. However, this greatly increases the number of parameters and the possibility of convergence problems.

Having a VPA approach in the general model would greatly improve the ability to compare VPAs with integrated models. Another improvement would be to allow VPA calculations for some fisheries that have highly variable selectivity to reduce the number of parameters, while allowing restrictions on the selectivity for other fisheries or surveys.

One option is to allow very flexible time-varying age-based selectivity in an integrated assessment to approximate the VPA approach. Such a selectivity setup is a good idea in general, so could satisfy this need. However, a true VPA approach may be more efficient because it has no parameters and may have an analytical form to estimate the fishing mortalities.

Should data-poor methods be integrated into the next generation general stock assessment models?

There are a number of data-poor methods advocated for use when data for standard stock assessment models is unavailable or inadequate. These methods use a variety of data types and simplifying assumptions. Often the inherent assumptions are not immediately apparent. The underlying dynamics for a data-poor method should be the same as those used for a standard stock assessment model since these describe standard population dynamics. Therefore, arguably, it should be possible to represent a data-poor method with a standard stock assessment model. If not, then the assumptions inherent in the data-poor method may be unrealistic. If implemented in a stock assessment model the assumptions will be transparent. Also, alternative assumptions or available data could easily be investigated.

On the U.S. West Coast the comparison of purpose-built data-poor methods with standard stock assessment models was not hampered by assumptions about the dynamics, but rather the parameter sampling algorithm that was used in the data-poor model with only 3 or 4 parameters. In the end, the purpose-built data-poor method was reasonably replicated using MCMC with an integrated model.

What system should be used to decide what features are included in the next generation model?

A wide range of features could be included in a next generation stock assessment program, using a variety of approaches to implement them. Some ideas may be poorly developed, some may have a low priority, and some may simply be not very good. The features should be appropriate, and well tested

and understood. Given limited resources, the features should be prioritized to determine which should be implemented first. However, the foundations need to be developed such that all possible future features can be implemented without a complete recoding of the foundations.

A system is needed to decide what features are included. This would probably involve a review panel to evaluate a recommended feature. Perhaps the concept would be recommended and if provisionally accepted, a full specification document would be produced and reviewed.

There may also be a need to have the capability to include experimental features, that may be limited in their distribution.

Any next generation assessment model should have some applications in mind when it is developed, some (if not all) of which should be stocks that are already being assessed using existing models. The features used in those existing applications would seem like a minimal set to include in the next generation model.

Does the general model need the capability to allow the distribution of length-at-age to change over time?

Most age-structured models assume that the distribution of length-at-age does not change over time. However, steep size-specific selectivity curves (e.g. escape gaps in pot fisheries and/or minimum legal size limits) and high exploitations rates can cause large changes in the distribution of length at age. The most flexible solution to allow for the length-distribution to change over time is to explicitly model both age and length. However, this greatly increases the computational demands. Several approximations have been developed (e.g. multiple growth platoons, modelling the moments of the distributions), but their application has been limited. It is not clear what approaches should be implemented in the general model. Explicitly modeling both age and length might require specific changes to the foundations of the model.

It is possible that this can't be answered until approaches that allow length-at-age to change over time are applied to a large set of stocks to see how much difference it makes and whether the fit to the data are substantially improved. This would be straightforward to do with a large set of models that use Stock Synthesis, it just hasn't been done yet.

How can size-structured models be included in the general model?

Most assessments are based on age-structured models. However, there are a large number of assessments based on size structured models. Size structured models are typically applied to species such as crabs and lobster that are difficult to age. However, they would also be useful for species for which the population processes are predominantly size-based. The general model could be developed to be fully size and age structured and the relevant partition (age or size) eliminated/integrated out of the specific application. However, this may add too much additional computational complexity. Alternatively, separate age and size-based models could be included in the code of the general model and the appropriate code used depending on whether the user chooses to use an age or a size-based

model. The other parts of the code will be used for each model type. Alternatively, a completely separate general size-based model could be developed, possibly sharing some of the subroutines from the age-based general model, where appropriate.

Does the general model need multi-species capabilities and should they be just technical interactions or include trophic interactions?

The ecosystem approach to fisheries management is becoming more popular and needs to be considered in stock assessment advice. Full ecosystem-based models are probably not practical in a general stock assessment program. However, multispecies models and models of intermediate complexity have been developed that are similar to standard stock assessment models. Technical interactions among species (different species caught in the same fishery) are much easier to implement than biological interactions (e.g. predation and competition). Biological interaction can be implemented in multiple ways ranging from simple indices of predation (perhaps implemented as a "fishery") to fully integrated multispecies models that fit to diet data. It is not clear how comprehensive the multispecies component of the model should be. However, if multispecies functionality is desirable, much time should be devoted to specifying the multispecies functionality before the foundations of the general model are developed to ensure that the desired functionality is feasible.

Should meta-analysis (Robin Hood) approaches be included in the general model?

Many stocks have limited data and assumptions need to be made to allow the development of an assessment and the consequent management advice. One approach is to share information from information-rich stocks with information-poor stocks. Typically, this has been done by meta-analysis of assessment results. However, doing the analysis outside the stock assessment model has some deficiencies in accurately representing the information and propagating the uncertainty. On the other hand, integrating multiple stocks and/or species into the same assessment model would greatly increase the complexity of the assessment. If the general model already has the capability of multi-species assessments, then adding the sharing of parameters makes sense.

Should the general model include automatic data weighting approaches for all data types?

Data weighting has become an important consideration in integrated fisheries stock assessment. Data weighting affects the relative influence of each dataset as well as the overall estimates of uncertainty. Similarly, the variance parameters of process error need to be determined. Ideally, the data weighting would be automatic. Unfortunately, the current approaches used for data weighting are less than adequate, often requiring ad hoc iterative procedures, not handling correlations well, need random effects/state-space modelling, and are not available for all data types. It is not clear what types of data weighting procedures should be available in the general model.

So long as data-weighting remains an active area of research, it may be necessary for a general model to include both automatic data weighting approaches as well as less efficient iterative procedures in order to compare the results of the alternative methods.

Is the inclusion of multivariate likelihood functions necessary for the index, index composition, and catch composition data.

Most, if not all, general stock models assume that likelihoods for data components are independent. However, some data are correlated (annual values of a CPUE index) and new methods may imply correlations among data types (e.g. simultaneous spatio-temporal modelling of CPUE and composition data to generate index, index composition, and catch composition data). Such data sets would be more appropriately fit using joint likelihood functions. This would require inputting the covariance matrices to use in the likelihood function. It is not clear whether using joint likelihood functions would improve the assessments.

What type of facility for including prior information should be available in the general model?

Bayesian analysis has become common in fisheries stock assessment due to the ability to include prior information and represent uncertainty. The move towards integrated analysis has eliminated some need for data-based priors, but priors are still an important part of many assessments, even those that use maximum (penalized) likelihood and don't conduct full Bayesian integration. Therefore, the facility to include prior information is important to have in a general model. There is some overlap and similarities between priors and data, and questions about the statistical rigor of many approaches in fisheries stock assessment (e.g. priors on parameters or derived quantities etc.).

However, some general approach to including prior information is needed. Decisions need to be made about which parameters or derived quantities priors can be applied to, and the forms of the priors. Specific attention might be needed to the correlation of prior information among parameters. For example, von Bertalanffy growth parameters estimated from an external analysis of age-length data will likely be correlated and this correlation should be propagated through the prior into the stock assessment for which the parameters will be updated from other information included in the stock assessment (e.g. the length composition data).

A general approach to allow priors with correlation among any model parameters might be desirable. This approach may be relatively straightforward to implement using a multivariate normal distribution, but the shape of the distribution may only be appropriate if the prior information is strong.

The FishLife project by Thorson et al. https://github.com/James-Thorson/FishLife provides bivariate predictions of life-history traits (such as M and K) for all marine fish species in the world. An option to include bivariate priors would facilitate the use of these predictions as parameter priors.

In a marine mammal application, Brandon et al. (2007) https://academic.oup.com/icesjms/article/64/6/1085/615665 showed that a bivariate prior was needed to avoid an incoherent joint prior distribution resulting from independent priors on two parameters.

However, such approaches seem unlikely to be adopted by more than a few people in the near term, so if this option is available, it would be important to not make it more complex to specify independent priors on individual parameters.

Should simulation and MSE features be included?

Simulation testing of stock assessment models and assumptions is a vital part of determining the appropriate model to use for a particular problem. This process requires simulation of both the population dynamics and the data gathered from the population. A related topic is MSE where the simulated population and data are used in a management loop to test a comprehensive management procedure that defines both the management action and the assessment methods. Some general packages have fully developed simulation capabilities, others have independent programs that are used, while others have both (e.g. SS and SS3sim). MSE is less well developed, but there are some general MSE packages that include a variety of assessment methods (e.g. Tom Carruthers's program). The obvious advantage of having simulation and MSE built into the general model is the convenience of having everything in one package and not needing to code the transfer of information from the estimation model to the simulation model and back. On the other hand, standalone simulators typically have more features and may allow for assumptions not included in the assessment model (e.g. data contamination, or different movement processes).

Operating models for MSE are often more complicated than the assessment model as they try to cover a wide range of sources of uncertainty, which suggests that an independent package for the simulator is preferable. Nevertheless, operating models should be conditioned on the data so that they are rational, which suggests integration into the general stock assessment model to force the good practice of using data-conditioned operating models. Both approaches could be taken, but this would result in duplication of effort. A decision should be made about which approach is to be used. Either approach should be adequate and it is not clear which approach would be the most advantageous. Nevertheless, the general model needs to include simulation and MSE features.

Ian: the inclusion of basic simulation and/or MSE features in a new platform does not limit anyone from developing independent simulators or MSE tools outside the model as well.

User interface and good practices defaults

Should all possible features be included or just those that are considered good practices?

A huge variety of features could be included in a stock assessment model. For example, growth curves could be linear, von Bertalanffy, Richards, Schnute, growth cessation, Brody, Gompertz, etc., and each of these could have different parameterizations that have benefits in specific situations. In some cases, a flexible model could be used to allow implementation of the different options (e.g. the Schnute model can represent several other growth models with the parameters fixed at appropriate values). It is good to have a choice, but users often do not have the experience or knowledge to know which feature to use in a specific application. Therefore, it may be wise to only include those features that are known to perform well and include advice on when to use a specific feature. Also, having many features or options for a specific feature makes the model and its use cumbersome. Therefore, if all possible features are included, the user interface needs to be such that the models use is not cumbersome no matter how many features are available.

If only good practices or some expanded “not bad” practices are implemented, then a formal approach is needed to decide what represents good practices to include in the next generation model. There will always be a need to define a system to decide what should be included in the model, but a best practices approach will be somewhat different.

Casal2 by default has “nothing” for the model. Every aspect of the model is determined by the user using the configuration file to define their models. Casal2 has a concept of different states where a model can be sanity checked (not yet used). Consideration has been given to the use of model templates that create a known model structure and allow the user to specify only the parameters they wish to modify.

Some consider that the only way we can learn which practices are good or bad is through comparisons using models where multiple approaches are available. Even known bad practices can be useful to have available for transition from models which had no other option. It may be more important than ensuring only good practices are available to provide warnings or finding other useful ways to provide guidance to users when they attempt to setup models in ways that are widely understood to be not good.

What kinds of comprehensive user interfaces are worth the effort?

Most general stock assessment models have rudimentary user interfaces often based simply on text files. Others have more developed user interfaces. For example, SS has a GUI as well as a text file-based user interface. The GUI requires more resources to develop and maintain, and updating the GUI is often delayed. The majority of users use the text file-based interface. However, SS also has an R-based output interface (R4SS) with a browser display option, which is commonly used.

It is not clear that a GUI interface is beneficial but developing a GUI should be considered. Adding large amounts of data (e.g. composition data) is often cumbersome in a GUI and finding options to turn them on and off can be difficult, which is probably why many assessment authors prefer text files to GUIs. Assessment authors also tend to be more familiar with text files than GUIs. Perhaps most importantly, GUIs can lead to errors and are difficult to automate.

MULTIFAN-CL has an R-based package (R4MFCL) for manipulating both the text-based input files and the output files. It is commonly used for configuring assessment options in the input text files and setting up a series of assessment runs. Using an R script with tools from an R package as the user interface has the advantage of repeatability and automation, which reduces error and increases the potential number of runs. It also provides guidance when an earlier assessment is updated. This kind of user interface is arguably the most useful for stock assessments that are to be used for management.

One benefit of a well-developed user interface is the inclusion of error checking. Given the large number of options in a general model, it is easy to make errors (e.g. forgetting to turn on or off an option or getting data misaligned). Error checks may reduce the risk of these errors.

Good user interfaces also make the model more accessible to inexperienced stock assessment authors. However, this can also be problematic because inexperienced users may make poor choices when

setting up a model, and are more likely to make errors in the setup. Consideration should be given to implementing good practice defaults in the user interface.

Given the wide adoption of R, the next generation model should have an R based package to view results and diagnostics.

Python should be considered for developing the user interface. It's a language that is gaining great uptake in the scientific community because of the ease of use and ease of finding help. It's also one of the core languages used in machine learning and AI/GPU based modeling systems.

Casal2 doesn't have a GUI, text files are used and an error system that identifies the file and line where the error occurred with a message of what the error was. The majority of use errors are handled automatically by the system through an inbuild parameter system (e.g. type conversion, invalid ranges, invalid values).

R is the current lingua franca of stock assessment scientists worldwide. Any model that doesn't have associated R functions to interact with it is not going to be widely used until someone inevitably develops an R package to fill that gap. However, there are many advantages to the use of plain text files as the fundamental input/output format to allow flexibility in the tools used to interact with the model, whether they are read or written by R, a GUI, Excel, or any other tool. Many assessment authors use text files rather than the user interfaces available for the general model to set up the model. However, the same authors typically use the (often R based) user interfaces to view results.

Should there be a comprehensive error checking system?

Due to the complex nature of general model and the vast range of features available, it is easy and common for assessment authors to make errors in setting up their models. It is highly desirable that comprehensive error checking systems are available to warn the user if the setup has potential errors that would affect the assessment results.

Casal2 has multiple levels of error checking available. A warning is displayed to the user when something may be a mistake. This does not stop the model execution but is displayed to the user. When errors occur from bad user input, Casal2 attempts to collect as many as possible before displaying them to the user. This prevents a 1 error, 1 run, 1 error, 1 run cycle. This makes it much easier and quicker for a user to diagnose and correct multiple errors. Casal2 has a Fatal error where the system must immediately quit and will notify the user of the problem. This is used when Casal2 Is unable to collect multiple errors before finishing. Casal2 has a CODE_ERROR output. This is where something in the code is not consistent and is a developer bug. A message containing information about the failure and a message to contact the developers is displayed.

Coordination, project planning, and funding.

Is a next generation model necessary (as opposed to continued development of existing models)?

There are several general stock assessment models available that are consistently used for applications and are continuously being improved with additional features added. However, there are some features that are not available in most general models (e,g, integration over random effects using la Place approximation) that may require substantial coding. Therefore, we are at the juncture where we need to make a decision about whether to modify existing packages or start programming a new package from scratch. There are advantages and disadvantages to both. Further developing the existing packages avoids reprogramming all the currently available features, avoids users to having to learn how to use a new system, and retains confidence in the package that is already known. However, many current packages are poorly coded because they have been written by stock assessment scientists and not computer programmers, so recoding using coding standards offers many advantages. It addition, it is not certain if all the new desired features can be integrated into existing models. If resources can be combined and the necessary time spent planning the project, developing a new generation stock assessment model is probably the best way forward.

What factors have influenced the rate of adoption of existing models and what lessons can be learned from those experiences?

Among the different general model packages there has been a different rate of adoption. Some packages like Stock Synthesis have been widely adopted, while others like Coleriane disappeared. There are a variety of reasons some packages are more widely adopted including available features, developers responsiveness to modifications, availability of training, promotion, developers personal network, publications, ease of use. These should all be taken into consideration because widespread use of a package has a large number of benefits including What factors have influenced the rate of adoption of existing models and what lessons can be learned from those experiences? (In the case of Stock Synthesis, widespread use has had clear benefit in improving the model in countless ways including new ideas for features, bug reporting, justification of the investment in development, etc.

Is it necessary to be able to replicate results from existing stock assessment models to provide a bridge for users trying to make the transition to a new model?

Intuitively, these should be a requirement for any general model. In fact, on the U.S. West Coast, this is required in the terms of reference for stock assessments. However, not all existing stock assessment applications make sense under current knowledge or thinking. Therefore, the standard structure and features included in the next generation general model may not include all the possible options to represent existing applications. Adding these options would require additional work and possible overhead. This question is related to other focus questions that address whether only good practices should be included in the general model or options for “bad” or not so good practices also be included.

Should available resources be combined and dedicated to a single next generation model or should several next generation models be constructed by different groups?

Currently there are several general stock assessment packages available developed by different groups. These packages have features in common, but also differ in important ways. The differences are due to several reasons including the needs of the stocks being assessed, and the expertise and interests of the developers. For example, MULTIFAN-CL integrates tagging data because surveys are not available for tuna stocks and tagging data has become an important data type for some tuna stocks. Integrating tagging data into a stock assessment is complex and therefore only a few programs have this feature. However, any next generation model should have this capability to be globally applicable. Similarly, random effects, which is a dominant feature of SAM, is a vital feature of any next generation model. The generality of a package such as SS is also highly desirable. Therefore, should the resources from SS, MULTIFAN-CL, SAM, and CASAL2 be combined to develop a model that is general, integrates tagging data (note that 3 of the 4 packages already do), and uses random effects, or should each of these programs be further developed to include the other features.

Despite the advantages of pooling resources in a single project, there are some advantages of multiple independent projects. Having multiple projects reduces the possibility of a fundamental flaw in the initial development of a project that may preclude an important feature in the future, since the flaw may not be in all projects. Also, developers may have different approaches and ideas leading to discovery of unique ideas that may not occur in a single project where creativity may be restricted by the thinking of the main developer. Also, cross comparison of different projects may help identify unknown specification or coding errors.

One approach may be to have a single next generation model, while encouraging independent projects based either on custom code or experimental branches of the main code. Once the ideas have been fully developed they can then be considered for inclusion in the main code.

Casal2 is working to be a hugely dynamic, easy to use, easy to fix and easy to conduct research with platform.

How should the next generation model be funded?

Acquiring adequate funding is vital for the success of any project to develop the next generation stock assessment model. Optimally, all funding would be available from a single reliable source. However, this may not be possible. Any other funding scheme has some risks that the funding does not become available or is delayed. Funding may need to be obtained from multiple sources and/or on an annual basis. Whatever the funding scheme, a rigorous budget would need to be developed to ensure that adequate funding is acquired.

Long-term ongoing funding will need to be secured, because general stock assessment models will always need maintenance and further development.

What project management system and software should be used to develop the next generation model?

Software projects require good management to ensure their success. For example, handling multiple developers, version control, experimental branches, committing changes, error checks, etc all need to dealt with. There are different approaches to software project management available and an appropriate one should be used.

There is no right answer to this, because there are many different tools. The system chosen shouldn't create friction for collaborators. It should be quick and easy for people who want to contribute to join. And everything possible should be automated. Casal2 runs 30 different build profiles when code is committed to the master branch. This is 15 Windows and 15 Linux builds.

Is there still a need for custom coded assessments if a comprehensive general model is available?

Today's stock assessments are a mixture of general model applications and custom code. Despite general model applications steadily becoming a higher proportion of the assessments, it is unlikely that custom coded assessments will be eliminated in the near future. A general model will never include all possible features, and the development and evaluation of new features may be easier in simplified code than in the code developed for the general model. Nevertheless, appropriate coding design may facilitate the development and testing of custom features within the general model code. So, although it is unlikely that custom coded assessment will completely disappear, there is potential for them to effectively disappear for official management use and to be relegated to the realms of research, if the general model is well supported and the additional of new features is rapid.

It should be noted that some of the differences between a general model and custom code are due to assumptions that are not necessarily the most appropriate, but more appropriate specification is available in the general model.

Custom code should not impact the model's capabilities. This is entirely dependent on how your software architecture is applied. Casal2 can have custom code added with no negative impacts on existing models or configurations.

How can we learn from other projects?

The whole world is now based on computer programs and therefore there is a large amount of experience out there on how to efficiently run a coding project. We should learn from this experience to ensure the efficient use of funds and to avoid any pitfalls. It might be best to focus on projects in the natural science fields. Computer modeling projects for weather and hurricane forecasting might be specifically relevant and well developed.

Non-scientific software should be considered. The domain for software isn't as important as the success the developers have had in building a pipeline for development/release and collaboration.