Paper OS04

Guiding the direction of open-source development for faster end-to-end HTA submissions

Christian Haargaard Olsen, Novo Nordisk A/S, Copenhagen, Denmark
Stephen Mc Cawille, Daichii Sankyo Europe GmbH, Omagh, Co. Tyrone, Northern Ireland

Phuse EU Connect - Hamburg - November 2025

Resume

EU HTA [1] presents a new challenge for biostatistics departments in the pharmaceutical industry. Requirements are strict and timelines will be short. Because the rules and requirements are similar for all companies, it is a good opportunity to develop the required tools together. However, the tools needed will need to cover a wide array of functionality, to comprise a full end to end solution. Seeing as individual companies will have an incentive to develop tools based on their internal processes and lagging tooling, it is easy to imagine both gaps and overlaps in functionality.
To address this challenge effectively, we need a coordinated approach that avoids duplication while ensuring comprehensive coverage. Based on modular design ideas we propose to co-develop a blueprint for the end-to-end pipeline, with a strict focus on standardized interfaces. This will allow different actors to build the parts that they feel provides the largest value for them, while ensuring that we will be converging on and end-to-end solution.

Introduction

The new EU HTA legislation[8] aims to streamline the process of seeking reimbursement in Europe while integrating with the process for regulatory approval to enable faster access to medication for patients.
This is a welcome ambition, as there is currently large discrepancies between how countries handle the reimbursement process. However, it is also a challenge, because the scope of the process is much larger than previously, on a stricter timeline, and with few options for front loading analyses as information on populations, indication, comparators and outcomes (combined, abbreviated as PICOs) of interest will be available very late.
The short time, between when the analysis requests are available and when the results should be delivered, puts heavy requirements on how the analyses are carried out. In reality one will need to standardize and automate as much as possible, to allow special attention to more complicated derivations and analyses. In addition, the planned strategy for most companies is to guesstimate PICOs to allow front loading analysis activities.
Current statistical analyses for HTA are often adaptations of the regulatory submission, or they are done using a tool specialized for the specific type of analysis at hand. Neither of these approaches will be efficient for EU HTA. We need something more automated and dynamic.

Since the entire industry is facing these new requirements, an obvious idea is that everyone could work together to produce tools that can support these new requirements. An earlier successful example of collaboration is the Pharmaverse network [5] and the R packages it curates. One such package is `admiral` [12] which was developed in a cross company collaboration between Roche and GSK.
An open source collaboration could be an attractive approach to building an end-to-end solution for HTA. However, both the technical difficulty and the collaboration presents many challenges:

- It is a large project. Many resources will be needed.
- It is a complex project. We need to manage many and complex dependencies.
- It is difficult to argue to allocate resources to something with unclear and uncertain results.
- We need to organize and break down the work.
- We need to decide what is included and what isn't.

Background

Before we dive into our proposed approach to undertake end-to-end HTA automation, we will provide a bit of background information on open source and open source collaboration, how programming is evolving in pharmaceutical statistics and some key ideas from modular design theory.

Open source collaboration

At its core open source means that the source code for a program is available for customization and tinkering. However, giving access to and actively sharing source code, has also given rise to new tools and methods for doing collaborative development. An example of a piece of software that was developed as open-source is the operating system Linux, which powers the servers behind 58% of all websites. An example of a tool, with related workflows, that support open source collaboration is git (in itself an open source project). While originally developed to facilitate the collaboration on and development of Linux, it has since become the cornerstone of online collaborative development, ensuring transparency in and book keeping of code changes done by different people when collaborating.
The Pharmaverse network[5] presents a combination of many open-source tools: developed individually and then published on the platform, or developed in collaboration in public through the platform. While it is clear from the success of Pharmaverse that open-source is a viable strategy for collaborating on developing tools, the topic often prompts a myriad of questions when mentioned in clinical statistical programming, like:

- Who is responsible for the software?
- What if the maintainers stop updating it?
- How can I ensure the quality?

These and many other foundational questions about open-source are addressed in the book "Open Source Technology in Clinical Data Analytics" [7].

Building a collaboration

In addition to the basic questions about responsibility, support and quality around the use of open source software, other questions arise when attempting to form a community to develop software.

- How do we motivate the community to participate actively?
- How do we make it clear what the value is in building this?
- How do we make it easy to participate?
- How do we make sure everyone is pulling in the same direction?

These don't have general and clear answers, and the approach taken will depend on the project and the context.

Building complex solutions

Another type of challenge in building systems such as an end-to-end system for EU HTA, are due to the complexity of the system.

- How can we break down the pipeline into manageable pieces?
- How do we ensure that the code individual A builds is compatible with what B builds?
- How will inner changes to one piece of the code affect the functionality of another piece?
- How do we ensure the quality of new updates? And that they don't have unintended effects?

These are not exclusive to inter-company collaborations, but will need to be addressed. One approach that can guide decisions to alleviate some of these challenges is `modular design`.

Modular design

Modular design is the idea of breaking down a system into smaller pieces, called modules. Breaking the system down into smaller pieces, allows for a clearer focus when developing each module. The idea originated in the physical world, where modular design has helped decrease the cost and increase efficiency by allowing piece-wise production and evolution. An example from the physical world is a car platform, where one or multiple manufacturers decided on a design and specification for the under-body as well as suspension, but leave the remaining parts of the car to be decided by the maker and model [4]. Sharing a platform allows sharing of components across multiple vehicles, thus lowering costs and enhancing operational efficiency. Figure 1 shows how the common platform supports different car types. An interesting source for learning about which cars belong to the same platform is Wikipedia, that among others hosts an overview of all of the Volkswagen Groups platforms[15].
Within software engineering the idea of modular design is a typical strategy for breaking down complex systems into smaller parts that can be developed individually.
For this strategy to be successful in software as well as elsewhere, there are a number of considerations to take[14].

Independence and encapsulation Each module should be self contained, have a clear purpose and internal logic. The modules inner workings should work independently of how other module operate internally.

Figure 1: Williams Advanced Engineering (WAE) and Italdesign are cooperating on a modular platform for building a wide variety of high-end electric cars.

Standardized interfaces and seamless integration The interfaces for in- and output for the module should be clearly defined, to allow for seamless integration with other modules.

Re-usability and scalability Reusing existing components provides consistency and reduces cost.

Low coupling and high cohesion Modules should be designed with low coupling to other modules. That is the inner workings of one module should be as independent as possible of the inner working of others. High cohesion inside the module reflects that everything inside the module should be working towards the same functionality.

Doing modular design through these design principles will make it possible to

- Divide a large task into smaller modules
- Develop each module individually and in parallel
- Apply test-driven development
- Change or replace a module without jeopardizing the overall system behavior
- Add additional modules without jeopardizing the overall functionality

While this sounds promising, there is no free lunch. For the components to integrate seamlessly, the scope of each module and the connecting interface between modules need to be precisely prescribed.

Programming in statistics

Programming in the pharmaceutical industry traditionally served the purpose of simple presentations of clinical data. Transparency was ensured by creating one program/script per output. This made it easy to check the code behind a table or figure.

However, over time the amount of analyses produced have grown significantly. To ensure consistency between so many analyses, standardization of formats for intermediate data sets have been implemented through the CDISC organization in the form of SDTM and ADAM. This exemplifies a move away from ”one program - one output” towards a pipeline philosophy where common calculations are shared, and where the data flows only separates when data is handled differently. Recently CDISC has released the `Analysis Results Standard` (ARS) [2] as an evolution of the Analysis Results Metadata into a dataset that will make the data machine readable an support automated analysis and presentation.
The increase in the complexity of statistical programming, now also means that the requirements for the programmers are changing. In addition to running an analysis or presenting data, we are now also expected to (and do) build tools that can streamline this process.

Our proposal

We propose to apply a modular design to the problem of building an end-to-end machinery for doing HTA analysis, and leveraging cross-industry collaboration such as the HTA Working group in R Consortium to map out requirements, specify required modules and their interfaces, and thus defining the blueprint for the system. A illustrative example of the mapping of the modularized pipeline is shown in Figure 2.

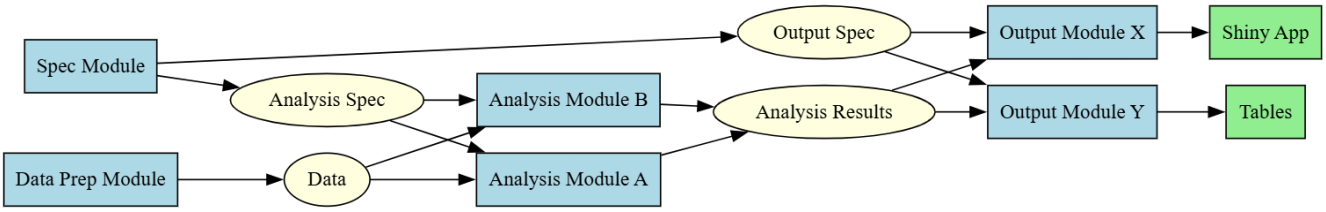

Figure 2: An illustration of what a modularized pipeline could look like. A specification module helps building the specification of analysis and it’s presentation. The information is made available through the ”Output Spec” and ”Analysis Spec” interfaces. Likewise, the ”Data Prep Module” ensures data derivations and access, and provides this information through the ”Data” interface - which in practice could just be exported data sets. Analyses Modules represents tools that can do different types of analyses, and leverages the analysis specification interfaces and data to get input for the analysis, and supplies the results through the Analysis Results interface. Finally different output modules can provide different types of presentations of the outputs, based on the information provided by the output spec and the analysis results.

Cross industry influence

The cross industry collaboration will ensure that when we map out the processes our system should cover, the result is one that is general to the field, and not one reflecting the inner processes of just one company. It will ensure that we have a shared vision we can work towards. Agreeing on the modules and their interfaces will ensure that what we build individually will integrate seamlessly and eventually comprise an end-to-end solution.

Well defined programming tasks

On a programming level, the modular approach will ensure that the task of programming a module is (more) well-defined task with a clear start and finish defined by interfaces. This removes

ambiguity and leaves the full power of decision making with the programmer building the module. It allows for automated testing of the individual module, but also for automated testing of their combination.

Value generating even without a full pipeline

On an implementation level, the modular approach means that as soon as one module is build we can begin to harvest the fruits of the investment - we need only transform our input to the right format. This means that if a company have an internal tool that can already do the task of one of the undeveloped modules, they will be able to - internally - connect their own tool to the pipeline and still get the benefits of the existing modules. For instance, a company with an existing SAS-based data preparation system could continue using it while adopting new analysis modules, as long as the output conforms to the agreed data interface specification.
One could also outsource the function of one module to a vendor, as long as the input and output satisfies the interfaces. Hence, it also provides a solution where outsourcing and automation and not a contradiction, but two pieces to the same puzzle.

Clear value proposition for development

At an organizational level, having a modular approach will make it easier to make an argument for why one should develop a certain tool/module. It is both clear what the task is, which makes resource estimation more reliable, and what the outcome will be, which makes the value clear. Breaking the problem down into modules changes the automation strategy from one big into the deep end and into a gradual descent through small steps.

Examples of modular thinking in pharmaceutical statistics

To illustrate how modular thinking already exists in pharmaceutical statistics, we examine two cases where these principles have successfully been applied.

SDTM and ADAM

With the introduction of the SDTM and ADAM models, data preparation was banished from the analysis programs, and anchored in different modules depending on their purpose. The SDTM and ADAM models themselves are in this case serving as the interface between the different modules: preparation of raw data as collected, manipulation and enhancement to prepare for analysis, and statistical analysis.
Programmers with experience in ADAM and SDTM programming will acknowledge that they would be able to work on ADAM programming, without worrying about how the SDTM data was prepared, as long as it followed the agreed upon scheme. And likewise for the programmers implementing an analysis: It does not matter technically how the ADAM data was derived, as long as it follows the agreed upon format. This serves as a good exemplification of *independence and encapsulation*, and to some degree of *standardized interfaces and seamless integration*, *Re-usability and scalability* and *Loose coupling and high cohesion*. ADAM, and to a small degree also SDTM, provides some flexibility in the interface, which impacts the last three principles. Figure 3 shows the CDISC workflow in a modules and interfaces visualization.

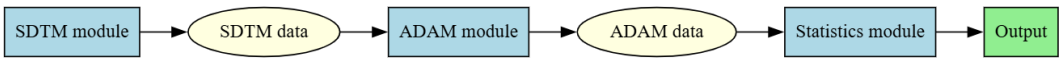

Figure 3: The use of SDTM and ADAM as interfaces in a programming pipeline with individual modules for preparation of raw data (SDTM), analysis ready data (ADAM) and outputs.

AMNOG pipeline

`ramnog` [6] is a suite of r packages created to facilitate efficient execution of analysis required for AMNOG submissions as part of the German reimbursement process. While the tool is build as a combination of several packages, and as such is organized in a modular fashion, one of us (McCawille) is exploring using it together with other existing packages in a pipeline setting.
This of course represents a more opportunistic and pragmatic approach than what we suggest as the overall strategy, but one that still exemplifies the modular thinking and enjoys some of the benefits of it. The idea is to leverage the data level AMNOG business rules included in the `ramnog` suite, the easy analysis possibilities of `CARDS` [10] to generate `ARD` output datasets, which can finally be presented using one or a combination of `tfrmt` [3], `gtsummary` [11], `cardinal` [9] and custom programming. The schematic is outlined in Figure 4.

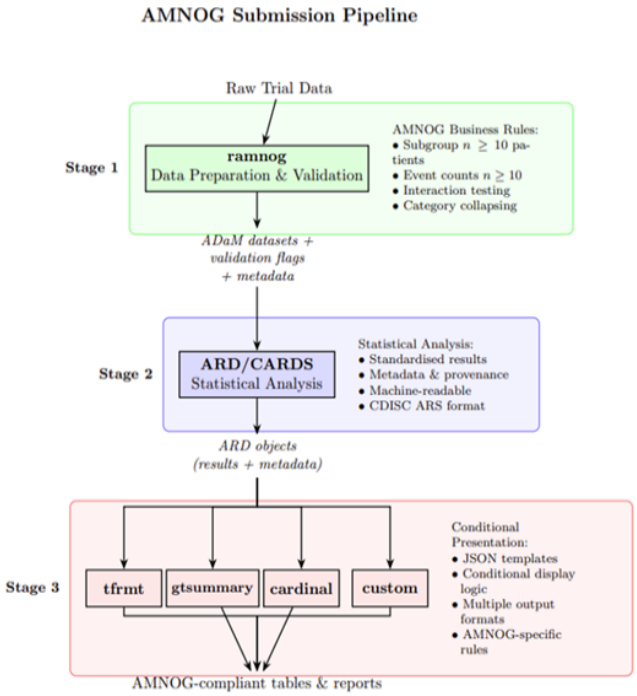

Figure 4: Modular architecture for AMNOG submissions with three stages connected by standardized interfaces. Stage 1 (`ramnog`) enforces AMNOG business rules during data preparation. Stage 2 (`ARD/CARDS`) performs statistical analysis and generates standardized results. Stage 3 offers multiple presentation options (`tfrmt`, `gtsummary`, `cardinal`, or custom tools) that consume standardized ARD objects and apply conditional display logic to generate AMNOG-compliant outputs.

Next steps

We have been involved in one of the sub-streams in the HTA working group in the R-consortium , working on mapping out the landscape around HTA. In terms of stakeholders, but also possibil-

ities for collaboration and on improving workflows and tooling for HTA. The area where we have progressed the most is within mapping out the processes involved in preparing for submission for EU HTA. This mapping was presented in a Webinar in June 2025[13], and we intent to leverage this mapping as the foundation to modularize the end-to-end pipeline.
We believe that the core principles of modularization can guide us in deciding what the best way to modularize is, we also acknowledge that implementation often come with unexpected dilemmas and a need for pragmatism.

Future tasks

In addition to deciding on the split between modules, there are plenty of other decisions and tasks downstream before we have a complete plan for an end-to-end analysis pipeline. One such tasks would be specifying the different interfaces - what information should they contain, and how do we propose they work. Deciding everything up front will make it easier for implementation, but comes with the downside that it is a time consuming task, and that it might miss some of the challenges that surface during implementation.
Furthermore, the suggested blueprint would describe how to split functionality and share information across the system, and while it will enable it will not ensure: quality of the components, traceability and automation. Just to mention a few.

Risks

While the modular approach offers significant advantages, several risks could impede the initiative's success.

Resources for blueprint development The most critical risk is insufficient backing for creating and maintaining the initial blueprint. Getting the right dissection between modules and ensuring a consistent coverage of the required processes is a comprehensive task. Without adequate support, we risk incomplete or uninformed modularization, incomplete interface specifications, which in turn will make it impossible to guarantee seamless integration during implementation.

Absence of central coordination Ongoing central coordination is essential when unanswered questions arise during implementation. Even with well-defined interfaces, ambiguities will emerge and specifications may need refinement. Without a recognized coordination mechanism, we risk interface drift, stalled development, and fragmentation into incompatible implementations.

Community Engagement and Sustainability Open-source collaboration requires sustained community engagement. Initial enthusiasm may wane over time and key contributors may change roles. Without active participation, modules risk becoming unmaintained or suffering quality degradation.

To mitigate these risks we suggest leveraging existing collaboration platforms, such as at the HTA working group in Rconsortium or Pharmaverse; establishing governance with power to decide the course in future questions, advise on implementation efforts, and suggest the most critical components for a minimum viable pipeline.

Conclusion

The pharmaceutical industry faces a significant challenge with EU HTA: strict requirements, short timelines, and the need for comprehensive end-to-end solutions. While open-source collaboration offers a promising path forward, the complexity and scale of building such systems present substantial hurdles.
We propose that modular design principles provide a framework to make this challenge tractable. By decomposing the end-to-end HTA pipeline into well-defined modules with standardized interfaces, we transform an overwhelming project into manageable, well-scoped tasks. This approach offers several key advantages:

Immediate value generation Companies can benefit from individual modules as they are completed, integrating them with existing tools as long as interfaces are respected.

Reduced coordination overhead Clear module boundaries allow parallel development by different organizations without requiring constant synchronization of internal implementation details.

Clearer business cases Well-defined modules make resource estimation more reliable and value propositions more transparent, facilitating organizational buy-in.

Flexibility The modular architecture accommodates different implementation strategies—whether through internal development, cross-company collaboration, or outsourcing—while maintaining system coherence.

The success of SDTM/ADAM standards demonstrates that modular thinking already resonates within pharmaceutical statistics. The HTA Working Group in the R Consortium provides an ideal forum to extend these principles to EU HTA workflows, ensuring the resulting blueprint reflects industry-wide needs.
Success requires sustained commitment to defining and maintaining interface standards, pragmatic decision-making during implementation, and ongoing community engagement. By adopting this modular approach the industry can converge on an end-to-end HTA solution through distributed effort, rather than waiting for a single monolithic system that will never materialize.

References

[1] European Commision. *Health technology assessment*. https://health.ec.europa.eu/health-technology-assessment_en. Accessed: 08OCT2025.

[2] Clincal Data Interchange Standards Consortium. *Analysis Results Standard*. https://www.cdisc.org/standards/foundational/analysis-results-standard. Accessed: 08OCT2025. 2024.

[3] Becca Krouse et al. *tfrmt*. Glaxosmith Klein Research & Development Limited and Atorus Research. URL: https://gsk-biostatistics.github.io/tfrmt/ (visited on 10/28/2025).

[4] Frank Markus. *What, Exactly, Is an Automotive Platform?* https://www.motortrend.com/features/what-is-an-automotive-platform. Accessed: 08OCT2025. 2021.

[5] *Pharmaverse*. https://pharmaverse.org/. Accessed: 08OCT2025.

[6] Matthew David Phelps et al. *ramnog - R packages for AMNOG analyses*. HTA pharma. URL: https://hta-pharma.github.io/ramnog/ (visited on 10/28/2025).

[7] PHUSE. *Open Source Technology in Clinical Data Analysis*. https://phuse-org.github.io/OSTCDA/. Accessed: 27OCT2025. 2022.

[8] "Regulation (EU) 2021/2282 of the European Parliament and of the Council of 15 December 2021 on health technology assessment and amending Directive 2011/24/EU (Text with EEA relevance)". In: *OJ* L 458/1 (22.12.2021).

[9] Pawel Rucki. *cardinal - A Harmonized Catalog of Pharmaceutical Tables, Listings and Graphs*. URL: https://pharmaverse.github.io/cardinal/ (visited on 10/28/2025).

[10] Daniel D. Sjoberg et al. *cards*. Insights engineering. URL: https://insightsengineering.github.io/cards/latest-tag/ (visited on 10/28/2025).

[11] Daniel D. Sjoberg et al. *gtsummary*. URL: https://www.danieldsjoberg.com/gtsummary/ (visited on 10/28/2025).

[12] Ben Straub et al. *admiral: ADaM in R Asset Library*. R package version 1.3.1.9008. 2025. URL: https://github.com/pharmaverse/admiral.

[13] Karolin Struck, Christian Haargaad Olsen, and Rose Hart. *R for Health Technology Assessment (HTA): Identifying Needs, Streamlining Processes, and Building Bridges*. Rconsortium. June 30, 2025. URL: https://r-consortium.org/webinars/r-for-health-technology-assessment-identifying-needs-streamlining-processes-and-building-bridges.html (visited on 10/28/2025).

[14] Visuresolutions. *What is modular design?* https://visuresolutions.com/plm-guide/modular-design/. Accessed: 08OCT2025.

[15] Wikipedia. *List of Volkswagen Group platforms*. https://en.wikipedia.org/wiki/List_of_Volkswagen_Group_platforms. Accessed: 28OCT2025.

Contact information

Comments and questions are encouraged and welcome. Please use below information to contact us.

Christian Haargaard Olsen

company Novo Nordisk A/S

email cino@novonordisk.com

linkedin https://www.linkedin.com/in/christianhaargaard/

Stephen Mc Cawille

company Daichii Sankyo Europe GmbH

email stephen.mccawille@daiichisankyo.com

linkedin https://www.linkedin.com/in/stephen-mc-cawille-1051874b/