FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Visualizing the LISA Global Fit: Architecture for High-Dimensional Bayesian Spaces

Pedro Miguel Martins Romão

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Paulo Garcia

Second Supervisor: Prof. Fernando Cassola

July 17, 2026

Visualizing the LISA Global Fit: Architecture for High-Dimensional Bayesian Spaces

Pedro Miguel Martins Romão

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Gabriel David
Examiner: Prof. André Moitinho
Supervisor: Prof. Paulo Garcia

July 17, 2026

Resumo

Introdução: A missão LISA, com lançamento previsto para 2037, irá observar um fluxo contínuo de milhões de sinais de ondas gravitacionais sobrepostos na banda dos milihertz. Separar as fontes individuais a partir desse fluxo é o trabalho do Global Fit, uma grande cadeia de inferência bayesiana executada em centros de computação distribuídos. O seu resultado é difícil de ler porque o número de fontes recuperadas muda de uma amostra para a seguinte. Os cientistas dependem hoje de scripts de visualização improvisados e frágeis, enquanto as ferramentas de uso geral não compreendem estes catálogos astrofísicos.

Objetivos: Definir o que uma ferramenta destas tem de fazer e conceber uma arquitetura de referência que transforme o resultado do Global Fit em diagnósticos interativos e rigorosos, capazes de escalar para o tamanho total dos dados da missão.

Métodos: O trabalho segue uma abordagem de Design Science Research, com requisitos obtidos através do processo de Engenharia de Sistemas da NASA e enquadrados pelas normas de software ECSS usadas em missões espaciais. A arquitetura mantém em camadas separadas a leitura de dados, o modelo científico e o desenho dos gráficos, esconde os vários formatos de armazenamento da cadeia atrás de uma única interface e representa cada amostra com o seu número real de fontes. É construída como uma biblioteca Python modular e testada no conjunto de dados LDC2a Sangria, cujas fontes injetadas conhecidas confirmam que os gráficos estão corretos.

Resultados: A biblioteca produz mapas do céu, gráficos de canto, gráficos de traço e verificações de convergência, reconstrução de sinal e vistas da F-statistic, que se combinam num painel de monitorização proposto. Ao reduzir as regiões densas a um campo de densidade e ao ler o catálogo por blocos, mantém a memória máxima estável perto de 322 MiB, desde alguns milhares de fontes até aos 26 milhões do catálogo real de Sangria, ao passo que carregar todas as fontes cresce até cerca de quatro gigabytes para um milhão. Todos os requisitos são cumpridos, ficando as medições de tempo e de *frame rate* para depois.

Conclusão: O trabalho dá à missão uma base de engenharia para as suas ferramentas de visualização, substituindo scripts frágeis por software que escala, uma década antes do lançamento. Novos tipos de fonte, como as Massive Black Hole Binaries (MBHBs), estendem-na adicionando apenas um leitor para cada. Um estudo de usabilidade com físicos fica para trabalho futuro. À medida que a missão amadurece, candidatos mais acessíveis deverão tornar esses testes viáveis.

Abstract

Introduction: The Laser Interferometer Space Antenna (LISA) mission, scheduled for launch in 2037, will observe a continuous stream of millions of overlapping gravitational wave signals in the millihertz band. Separating the individual sources from that stream is the work of the Global Fit, a large Bayesian inference pipeline run across distributed computing centres. Its output is hard to read because the number of recovered sources changes from one sample to the next. Physicists currently rely on fragile one-off plotting scripts, while general purpose tools do not understand these astrophysical catalogs.

Objectives: To set out what such a tool must do and to design a reference architecture that turns the Global Fit output into accurate, interactive diagnostics that scale to the full size of the mission data.

Methods: The work follows a Design Science Research approach, with requirements drawn through the NASA Systems Engineering process and bounded by the ECSS software standards used on space missions. The architecture keeps data reading, the scientific model and the drawing of plots in separate layers, hides the pipeline's several storage formats behind one interface and represents each sample at its real number of sources. It is built as a modular Python library and tested on the LDC2a Sangria dataset, whose known injected sources confirm the plots are correct.

Results: The library produces sky maps, corner plots, trace and convergence diagnostics, signal reconstruction and F-statistic views, which combine into a proposed monitoring dashboard. By reducing dense regions to a density field and reading the catalog in blocks, it holds peak memory steady near 322 MiB from a few thousand sources up to the 26 million of the real Sangria catalog, where loading every source instead grows to about four gigabytes at one million. All requirements are met, with timing and frame-rate measurements left for later.

Conclusion: The work gives the mission an engineered foundation for its visualization tools, replacing fragile scripts with software that scales, a decade before launch. New source types such as Massive Black Hole Binaries (MBHBs) extend it by adding a single reader each. A usability study with physicists is left to future work. As the mission matures, more accessible candidates should make such tests feasible.

The United Nations Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a better and more sustainable future for all. It includes 17 goals (https://sdgs.un.org/goals) to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace and justice.

This thesis sits closest to three of these goals. The work is a software engineering contribution to the ground segment of an international science mission, so it bears on Goal 9, which concerns research infrastructure, innovation and resource-efficient systems. It also bears on Goal 17, which concerns international scientific cooperation and the sharing of knowledge and technology. Finally, it aligns with Goal 4, concerning quality education, by providing accessible, well-engineered open-source tools that lower the barrier to entry for students and new researchers. The contribution is indirect rather than social. It does not act on people or the environment directly but strengthens the tooling and educational resources a scientific collaboration depends on, which is where its effect on the goals lies.

The specific Sustainable Development Goals addressed have the following names:

SDG 4 Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 17 Strengthen the means of implementation and revitalize the Global Partnership for Sustainable Development

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.5	Replaces the ad hoc plotting scripts the Global Fit is read through today with an engineered, tested visualization library, so physicists can verify convergence and interpret the trans-dimensional posterior without rebuilding tools for each run. This raises the research capability of the Laser Interferometer Space Antenna (LISA) ground segment a decade ahead of launch.	A passing automated test suite, including an architecture test that fails on any forbidden cross-layer import. Locating a non converging chain through the drill-down rather than through separate diagnostic scripts run by hand.
	9.4	The memory-efficient bounded streaming render keeps peak memory set by the processing block size rather than the source count, so a mission scale catalog is read within the memory available on a single workstation rather than through High-Performance Computing (HPC) time or a full terabyte transfer.	Peak resident memory held flat at about 322 MiB from 1,000 synthetic sources up to 26,000,000 real Sangria sources on the streaming sky map (Chapter 6), against an object path that reaches about 4 GB at 1,000,000.
	9.1	The architecture follows the European Cooperation for Space Standardization (ECSS) software engineering discipline with low coupling, high cohesion and requirements traceability, so the tool stays maintainable and verifiable across a mission whose lifetime runs to decades. New source types or upstream format changes are absorbed behind the repository interface rather than rewritten.	Zero infrastructure imports in the Core Domain, enforced by the architecture test. A format change reaching exactly 1 adapter while the domain and visualization layers stay unchanged (NFR-01, NFR-04).
4	4.4	By replacing undocumented, ad hoc scripts with a documented, cleanly architected open-source library, the project lowers the barrier to entry for students and junior researchers joining the LISA consortium. It serves as both a pedagogical tool for understanding trans-dimensional inference and a reference for modern software engineering practices in astrophysics.	The repository includes comprehensive documentation, explicit environment locking and type hinting, adhering to the standard practices for scientific computing (Wilson et al. 2016) to support onboarding and independent study.
17	17.6	The library is open source and installs through `pip` with no dependency on a specific HPC environment, so any member of the LISA consortium across European Space Agency (ESA), National	A self-contained, pip-installable package free of HPC-specific or version-control-only runtime dependencies, checked by an automated test (NFR-03).

Acknowledgements

I want to express my deepest gratitude to everyone who supported me throughout the journey of completing this dissertation.

First and foremost, I am sincerely thankful to my supervisor, Prof. Paulo Garcia, whose expertise, unwavering support and patience were invaluable at every stage of this work. I am equally grateful to my co-supervisor, Prof. Fernando Cassola, whose thoughtful advice always made things clearer. I would also like to thank Francisco Pimenta for explaining complex concepts in such an understandable way.

My family deserves my heartfelt thanks: my father, mother and brother have always been a source of encouragement and strength, as they were once again throughout this work.

I would like to thank the friends I made along the way, especially at Formula Student FEUP, an experience that demanded a lot but to this day brings me a sense of fulfilment and pride in what we achieved. Their companionship made the university experience far more enjoyable than I could ever have imagined.

To my dear Diana, thank you for being by my side throughout all of this. You not only made the process more tolerable but, above all, made every one of my days better.

To all of you, thank you for making this journey possible.

Pedro Miguel Martins Romão

Declaração de Honra

Declaro que a presente dissertação é de minha autoria e não foi previamente apresentada e avaliada nesta ou em outra instituição. As referências a outros autores (afirmações, ideias, pensamentos), assim como a utilização de trabalhos publicados respeitam escrupulosamente as regras da atribuição, e encontram-se devidamente indicadas no texto e nas referências bibliográficas, de acordo com as normas de referenciação. Tenho consciência de que a prática de plágio ou de autoplágio constituem ilícitos académicos, conforme estabelecido no Código de Ética da U.Porto.

Declaro, ainda, que utilizei ferramentas de Inteligência Artificial generativa na realização de parte(s) da presente dissertação. Essa utilização e os seus resultados estão devidamente assinalados na declaração estruturada abaixo e foram objeto de cuidada verificação para deteção de erros, imprecisões, atribuições infundadas ou outras formas de enviesamento de conteúdos e argumentos, bem como de determinação de autoria.

Declaração estruturada de uso de IA
As tarefas apoiadas por IA são identificadas segundo a taxonomia de referência GAIDeT (Generative AI Delegation Taxonomy), na linha das orientações da U.Porto para o uso responsável de IA generativa no contexto académico.

Responsável pelo trabalho: Pedro Romão.

Ferramenta(s) e período: Google Gemini 3.5 e Claude Sonnet 4.6 (Anthropic), utilizados ao longo da preparação da dissertação.

Âmbito do uso: apoio auxiliar à execução, sobretudo na aceleração do desenvolvimento da biblioteca. A conceção, as decisões, os argumentos e os resultados mantiveram-se sob responsabilidade do autor.

Tarefas delegadas: *escrita e edição* (revisão linguística, clareza e harmonização terminológica ao longo dos capítulos); *desenvolvimento de software* (geração de código auxiliar, sugestões de refatoração e apoio ao debugging durante a implementação da biblioteca, acelerando significativamente o desenvolvimento, sempre sob revisão e teste do autor); *visuais e multimédia* (refinamento dos diagramas de arquitetura e do modelo de domínio: títulos, dimensão, legibilidade e ligações entre componentes); *supervisão e controlo de qualidade* (deteção de inconsistências, verificação do estilo de citação e harmonização da formatação de tabelas).

Partes afetadas: revisão linguística geral e harmonização do texto em todos os capítulos; figuras de arquitetura e do modelo de domínio (Capítulo 4); a implementação da biblioteca (Capítulo 5).

Verificações humanas efetuadas: todas as referências e citações foram confirmadas nas fontes originais, não tendo sido utilizada qualquer referência gerada pela IA; as afirmações técnicas, os números e os resultados foram validados contra a implementação e os dados reais do *pipeline*; o código gerado com apoio de IA foi revisto, integrado e validado pela suite de testes antes de ser incorporado.

Dados e confidencialidade: não foram introduzidos dados pessoais, sensíveis ou confidenciais em ferramentas de IA. O trabalho assenta em conjuntos de dados públicos (LISA Data Challenges) e em código da autoria do próprio.

Responsabilidade final: a conceptualização, a arquitetura de referência, a metodologia, a seleção da literatura e todas as conclusões são da exclusiva autoria do autor. A implementação da biblioteca foi desenvolvida pelo autor com o apoio de IA acima assinalado, assumindo o autor integral responsabilidade pelo conteúdo final.

A título ilustrativo, a interação seguiu pedidos do tipo:

- "Após esta alteração, indica todos os locais onde mencionei esta ideia e como os reformular para o texto ficar coerente." (harmonização entre capítulos)
- "Torna o tom mais académico e melhora a legibilidade, sem alterar o conteúdo nem o significado." (revisão linguística e de estilo)
- "Verifica esta afirmação contra o comportamento real do *pipeline* e assinala o que não for defensável." (verificação crítica antes de aceitar qualquer sugestão)

Porto, junho de 2026

Pedro Romão

Se queres ver o Mundo inteiro à tua altura
Tens de olhar para fora, sem esqueceres que dentro é que é o teu lugar

Jorge Palma, *Terra dos Sonhos*

Contents

List of Figures

List of Tables

List of Acronyms

LISA Laser Interferometer Space Antenna

LIGO Laser Interferometer Gravitational-Wave Observatory

GB Galactic Binary

HSI Human Systems Integration

SNR Signal-to-Noise Ratio

NASA National Aeronautics and Space Administration

MBHB Massive Black Hole Binary

MCMC Markov Chain Monte Carlo

PSD Power Spectral Density

DDPC Distributed Data Processing Centres

ECSS European Cooperation for Space Standardization

ESA European Space Agency

HPC High-Performance Computing

API Application Programming Interface

DDD Domain-Driven Design

HDF5 Hierarchical Data Format version 5

SLR Systematic Literature Review

Chapter 1

Introduction

1.1 Context

The ground segment of a space mission is the ground based infrastructure that controls the satellites and analyses the sensor data they return. This thesis sits in that segment. The mission it serves is Laser Interferometer Space Antenna (LISA), an European Space Agency (ESA) mission scheduled for launch in 2037 to detect low frequency gravitational waves from supermassive black hole mergers and compact binary systems (ESA 2025). LISA observes spacetime distortions directly and continuously, so its data stream arrives as millions of overlapping astrophysical signals buried in instrumental noise. Pulling individual sources back out of that stream is the work of the Bayesian inference pipelines collectively called the Global Fit (Deng et al. 2025).

The Distributed Data Processing Centres (DDPC) Consortium carries out this analysis across international computing networks, so the visualization architecture for the Global Fit has to reach into varied High-Performance Computing (HPC) environments to collect, parse and visualize hierarchical fits of multidimensional gravitational wave models. Other space agencies have met the same need. Anghelea et al. (2023) describe the Earth Observation Dashboard, a multi-agency effort that pulled heterogeneous NASA, ESA and JAXA data into a single visual standard so complex satellite observations could be read by a broad audience. LISA poses the same integration problem on a smaller scale, an ESA led mission with National Aeronautics and Space Administration (NASA) participation whose Global Fit is produced across distributed centres. The move from ad hoc scripts to engineered tooling has a direct precedent too: Pham and Liao (2008) report that operators of NASA's Deep Space Network inspected performance telemetry by hand until folding those manual routines into one dashboard removed the bottleneck.

The Global Fit sits in the same position today. The pipeline orchestrates trans-dimensional Markov Chain Monte Carlo (MCMC) sampling across distributed HPC networks, so its core codebase runs to over 20,400 lines of Python[1] maintained by tens of contributors, before external C/C++ dependencies like `lisabeta`. Because it is engineered for mathematical inference rather

1. Measured on the Global Fit pipeline repository, https://gitlab.in2p3.fr/lisa-apc/global-fit.

than interface design, it lacks the visualization architecture to make its outputs interpretable, leaving physicists to inspect them through fragile per run plotting scripts.

The primary deliverable of this thesis follows from that gap: a formally structured reference architecture for visualizing the Global Fit's trans-dimensional MCMC output (Sarikaya et al. 2019). This architecture serves as the primary Design Science Research (DSR) artifact and is validated through a modular Python visualization library[2] and a proposed dashboard layout, with a full web deployment left as a later direction (Basset 2024).

1.2 Motivation

The motivation is to let the physicists who run the Global Fit read and validate its output. What stands in the way is the scale of that output and the lack of tools built to read it. Pulling the mission's science out of its data means disentangling tens of thousands of superposed signals through trans-dimensional MCMC and block Gibbs sampling (Deng et al. 2025). The volumes that process moves are large. A single training dataset, LDC2a Sangria, occupies over 3 GB (Cécile Cavet 2022) and the current common dataset, Mojito Light, already spans 1.3 TB (LISA DDPC 2026a). Each uncompressed Global Fit submission of recovered sources and posteriors adds about 20 GB (LISA DDPC 2026b), while the mission expects around 6 GB of analysis products per day over a lifetime of at least four years (Amaro-Seoane et al. 2017), so the accumulated output grows into the terabytes. Standard plotting tools were never built for posterior distributions of this scale and dimensionality, so reading the result calls for a dedicated interface that synthesizes the source catalog and validates the fit, which means replacing the current fragile ad hoc plotting scripts with a properly engineered software architecture (Basset 2024).

The work also belongs to an international space mission collaboration whose ground segment software is developed under the European Cooperation for Space Standardization (ECSS) standards (European Cooperation for Space Standardization 2025), so this project aligns its engineering with those standards from the start to keep the library fit for later integration. That is a second motivation alongside the scale above, because even a tool that could handle the data volumes would still have to be engineered for low coupling, high cohesion and strict domain models to live in a codebase maintained across the collaboration's distributed centres.

1.3 Problem

The root of the problem is the nature of the Global Fit's output. The sampler separates the overlapping signals, but what it returns is a correlated, high dimensional posterior over a source population whose size changes from one draw to the next. General purpose plotting tools were built for isolated, low dimensional datasets that look nothing like it (Amaro-Seoane et al. 2017).

This leaves the library two core challenges. The first is a data ingestion layer that can deserialize the heavy, highly structured HDF5 datasets the heterogeneous HPC networks produce, where

2. Publicly available at https://github.com/PedroRomao3/lisaviz.

Massive Black Hole Binary (MBHB) and Galactic Binary (GB) signals overlap extensively (Deng et al. 2025). The second is a modular plotting layer that turns that raw data into interactive, scientifically accurate views while holding off the severe overdraw of mapping millions of sources (Basset 2024). Both must be met under the ECSS standards above, which is why adjacent fields facing comparable astronomical telemetry have turned to formally structured data exchange formats rather than ad hoc scripts (Gomes et al. 2024).

1.4 Research Question

This thesis addresses the following research questions:

1. What are the precise requirements and specifications for a visualization tool designed to interpret the LISA Global Fit MCMC outputs?
2. How can software engineering principles (such as Domain Driven Design) be applied to map the Global Fit's heterogeneous inference data into interactive, scientifically accurate visualizations?
3. What implementation strategies allow the visualization library to scale to mission sized datasets, comply with ECSS standards and serve as the plotting engine for future monitoring dashboards?

1.5 Hypothesis

The working hypothesis is that a decoupled architecture, separating data ingestion from rendering and realized as a (Python) library, can turn the Global Fit's trans-dimensional output into scientifically accurate views physicists can read and verify at full mission scale, so it replaces the fragile ad hoc scripts in use today and becomes a first step toward an analysis system that gets more science out of the mission (Basset 2024).

1.6 Contributions

The central contribution of this thesis is a reference architecture for visualizing the trans-dimensional Bayesian output of the LISA Global Fit. The architecture is the part meant to generalize. It is expressed in language neutral software engineering patterns, so it is not tied to the Python implementation shown here and could be realized in another stack or carried over to a similar problem where the number of inferred sources varies from one draw to the next. It is demonstrated at three levels that run from the most general design to its concrete realization. The architecture itself sets out how data ingestion, the domain model and rendering are separated, how heterogeneous storage formats are hidden behind a repository interface and how variable cardinality MCMC draws are represented directly as domain objects. A Python library then realizes it and is the existence

proof that the design is buildable and meets its requirements for decoupling, bounded memory and ingestion performance. A dashboard layout finally shows how that library would compose into an integrated interface, applying the visualization and human factors knowledge surveyed in Chapter 2. The library is what shows the architecture works and the layout is what shows it is useful, but the main result is the architecture.

1.7 Scope and Limitations

Following a formal systems engineering approach (NASA 2016), the project implements one source type in full rather than every source type in part, so the initial scope stays small enough to validate the core architecture before any horizontal scaling. The LISA band contains several source types, but the Global Fit as it stands resolves two of them, Massive Black Hole Binaries and Galactic Binaries. Extreme Mass Ratio Inspirals are present in the input data but are not yet resolved by the pipeline. The implementation in this thesis narrows further to Galactic Binaries alone, because they are the most numerous population (Cécile Cavet 2022), so they combine the worst overdraw with trans-dimensional cardinality, which makes them the hardest case for the visualization architecture rather than the easiest. The abstract repository interface is what makes this safe as a starting point. As later chapters discuss, further source types can be added iteratively without touching the core visualization domain.

Chapter 2

State of the Art

The core literature for this chapter was established through a Systematic Literature Review (Systematic Literature Review (SLR)) at the beginning of the project, which set the initial base of sources on data intensive dashboards, ground segment operations and scientific data pipelines. This base was then extended less formally as the work progressed, by following citations from those sources and adding newer or directly relevant work as it appeared.

2.1 The LISA Global-Fit Challenge

2.1.1 The Nature of LISA Telemetry

The LISA mission faces a data analysis problem that is fundamentally different from existing gravitational wave observatories like Laser Interferometer Gravitational-Wave Observatory (LIGO). Ground-based detectors typically search for transient, short duration events in the 10 Hz – 10 kHz band (Abbott et al. 2016). LISA, by contrast, will operate in the millihertz frequency band (0.1 mHz – 1 Hz), a range set by its 2.5 million km interferometer arms, where the signals are long lived and continuous (ESA 2025).

As outlined in the official L3 mission proposal (Amaro-Seoane et al. 2017), the telemetry analysis pipeline is structured into three chronological data levels, each produced continuously over the life of the mission:

- **Level 1 (L1) Data:** Consists of the calibrated science telemetry streams and Time Delay Interferometry (TDI) variables generated by the Science Operations Centre (SOC), producing about 600 MB of calibrated laser signal per day.
- **Level 2 (L2) Data:** Consists of intermediate waveform products, such as partially regressed signals obtained by progressively subtracting identified sources from the main stream, as well as rapid alerts for transient astronomical events generated by the Data Processing Centre (DPC).

- **Level 3 (L3) Data:** The ultimate scientific output of the mission, consisting of catalogues of identified sources and their corresponding multidimensional posterior parameter distributions. The processing of Level 2 and Level 3 data generates about 6 GB per day.

LISA's detector (Fig. 2.1) sees a dense superposition of thousands of simultaneous sources rather than a handful of isolated events. The LISA Data Challenges (Cécile Cavet 2022) illustrate this complexity, particularly the "Sangria" dataset (LDC2a), which simulates a realistic mission year containing about 30 million Ultra-Compact Galactic Binaries (UCGBs) alongside Massive Black Hole Binaries (MBHBs). More recently, the official Common Dataset 1 Light (Mojito Light) injects over 15.5 million Galactic Binaries into the telemetry (LISA DDPC 2026a). Both datasets put tens of millions of overlapping sources in the band, which is the density the mission software must handle.

While approximately 10,000 of these binaries are expected to be individually resolvable, the remaining millions form a stochastic "confusion noise" foreground that dominates the instrumental noise in the 0.2 mHz to 6 mHz band (Crowder and Cornish 2007). Which binaries fall on each side of that line is not known in advance, since whether a source rises above the foreground or sinks into it depends on the data, so the number of resolvable sources is itself unknown. The pipeline therefore has to infer that number together with the parameters of the sources, which is what makes the inference "trans-dimensional" (Deng et al. 2025).

The raw data volumes are substantial. While the legacy Sangria training dataset occupies over 3 GB as a single Hierarchical Data Format version 5 (HDF5) file (Cécile Cavet 2022), the complete un-downsampled Mojito dataset exceeds 1.3 TB (LISA DDPC 2026a). HDF5 has emerged as a standard mechanism for storing large quantities of numerical data, both in the Python scientific ecosystem and in space mission data handling more broadly, because it natively supports hierarchical organization, arbitrary metadata and partial I/O (Collette 2013). On the analysis side, each uncompressed Global Fit submission of recovered sources and posteriors runs to about 20 GB (LISA DDPC 2026b), while the mission plans for around 6 GB of analysis products per day (Amaro-Seoane et al. 2017), so years of accumulated data and repeated pipeline runs push the total into the terabytes. Data on this scale forces a decoupled, memory-efficient architecture for gravitational wave visual diagnostics.

2.1.2 The Modular Global-Fit Pipeline

To address this source confusion, the current standard approach is the "Global Fit", a Bayesian inference pipeline designed to disentangle these superposed signals. Basset notes the computational challenge is expected to be an order of magnitude heavier than the data processing for the recent ESA Euclid mission (Basset 2024).

The current architectural approach (Figure 2.2) relies on a Block-Gibbs sampling strategy (conceptualized as a "Wheel of Fortune"). The parameter space is split into blocks (e.g., GB, MBHB, Noise) and the pipeline iterates through them, updating one block while holding others

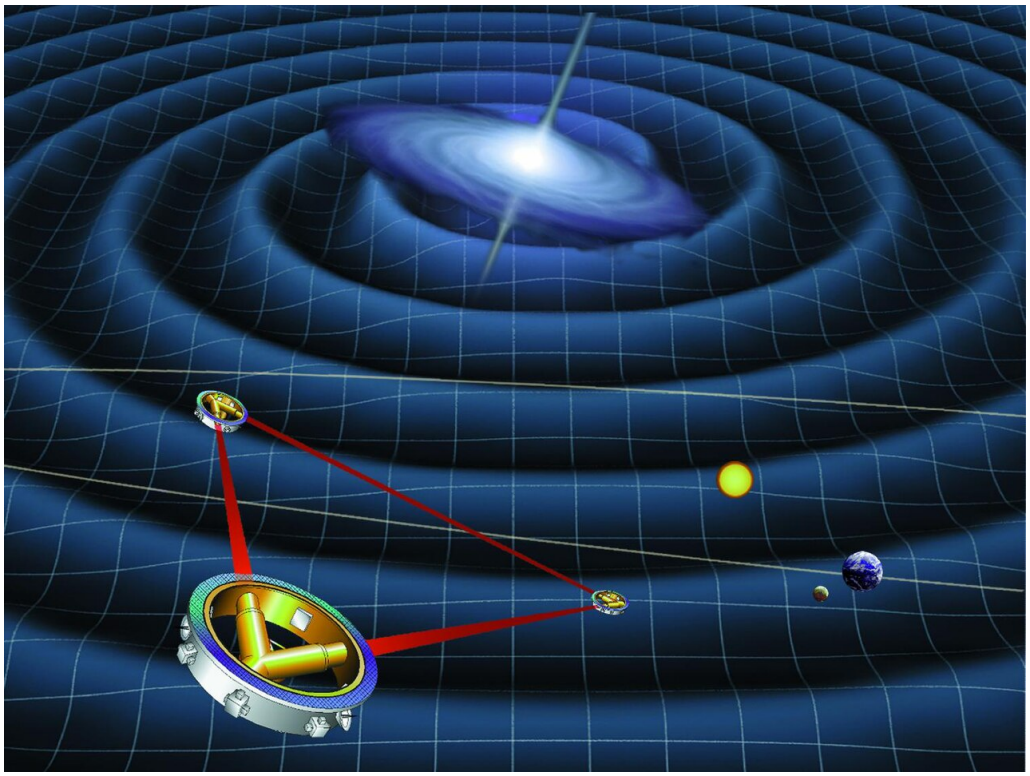

Figure 2.1: **Artist's impression of the LISA constellation.** The observatory consists of three spacecraft in an equilateral triangle configuration with 2.5 million km arm lengths. *Source: ESA/ATG medialab.*

fixed. This lets the blocks run across different computing centers, but it introduces significant complexity in orchestration and convergence monitoring (Deng et al. 2025).

2.1.3 Galactic Binary Sources and Parameters

Each Galactic Binary is fully characterized by eight physical parameters, summarized in Table 2.1.

These parameters fall into two categories (Crowder and Cornish 2007). Parameters 1–2 (frequency and its derivative) and 4–5 (sky coordinates) are commonly called *intrinsic* parameters because they determine the shape of the waveform template. Parameters 3, 6, 7 and 8 (amplitude, inclination, polarization and initial phase) are *extrinsic* parameters: they scale or shift the waveform but do not change its fundamental structure.

The pipeline does not search all eight parameters at once. The four extrinsic parameters enter the detector response linearly, so for any fixed choice of the intrinsic parameters their best fit values follow directly from a least squares solution rather than from a search. Substituting those values back into the likelihood collapses the initial eight parameter problem to the F-statistic, a score over the four intrinsic parameters alone: frequency, frequency derivative and the two sky coordinates. Evaluating it across a grid over that intrinsic space produces the F-statistic landscape, whose peaks mark candidate sources. Candidates then seed a parameter estimation MCMC that samples the full eight parameters, so the posterior carried forward is eight dimensional rather than

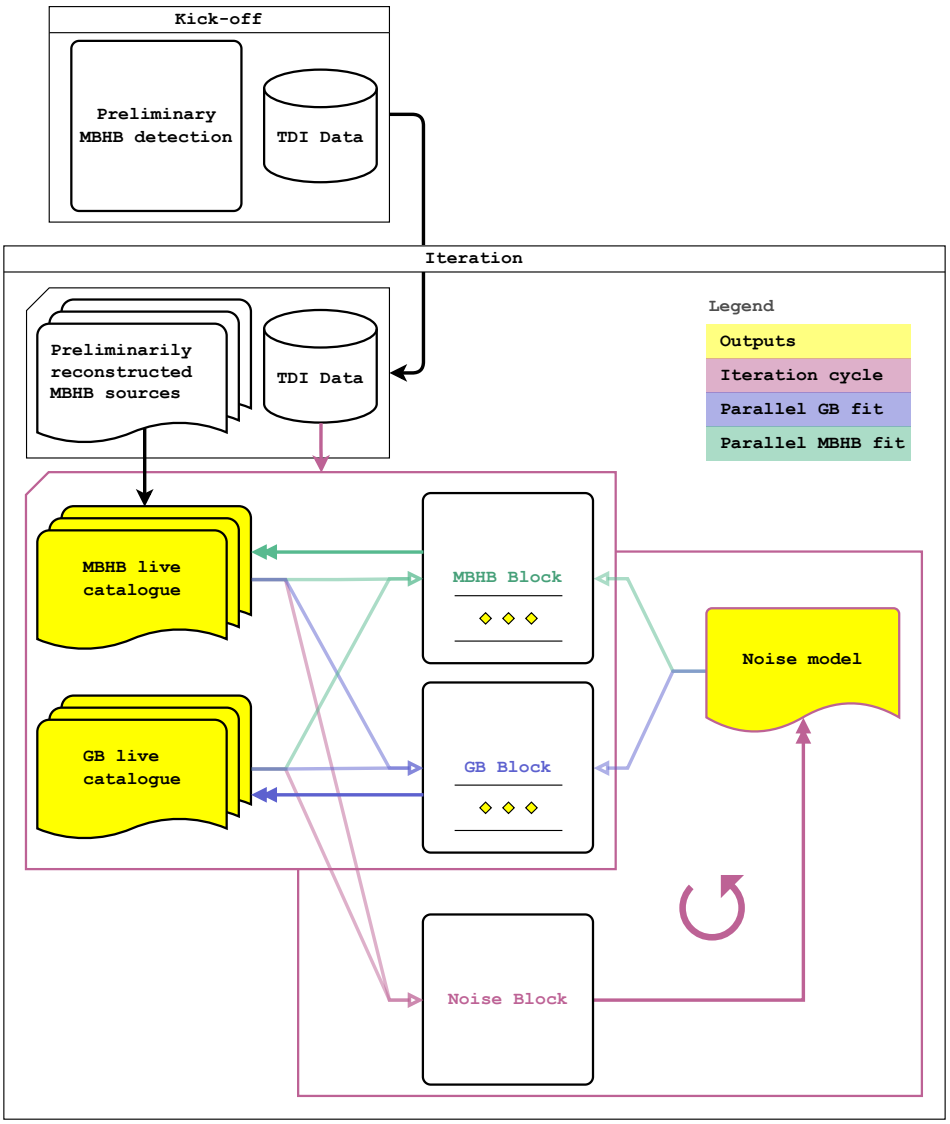

Figure 2.2: **Schematic of the Modular Global-Fit pipeline (the "Wheel of Fortune").** The system iterates through distinct signal blocks (e.g., GB, MBHB), where each block updates its parameters based on the residuals left by the others. *Adapted from Deng et al. (2025).*

four for each candidate. A separate trans-dimensional stage infers how many of these candidates the band actually holds. The resolved catalog records the surviving sources with all eight parameters. Visualizing these posteriors across potentially thousands of simultaneously resolved binaries, each with its own multi dimensional structure, is a hard visualization problem.

Figure 2.3 follows a single source through the stages. The F-statistic grid works in the 4 intrinsic parameters because the 4 extrinsic ones are fixed analytically by the least squares fit rather than sampled. The parameter estimation MCMC that follows samples all 8, so the resolved catalog records the full 8 dimensional description rather than the 4 the grid began with.

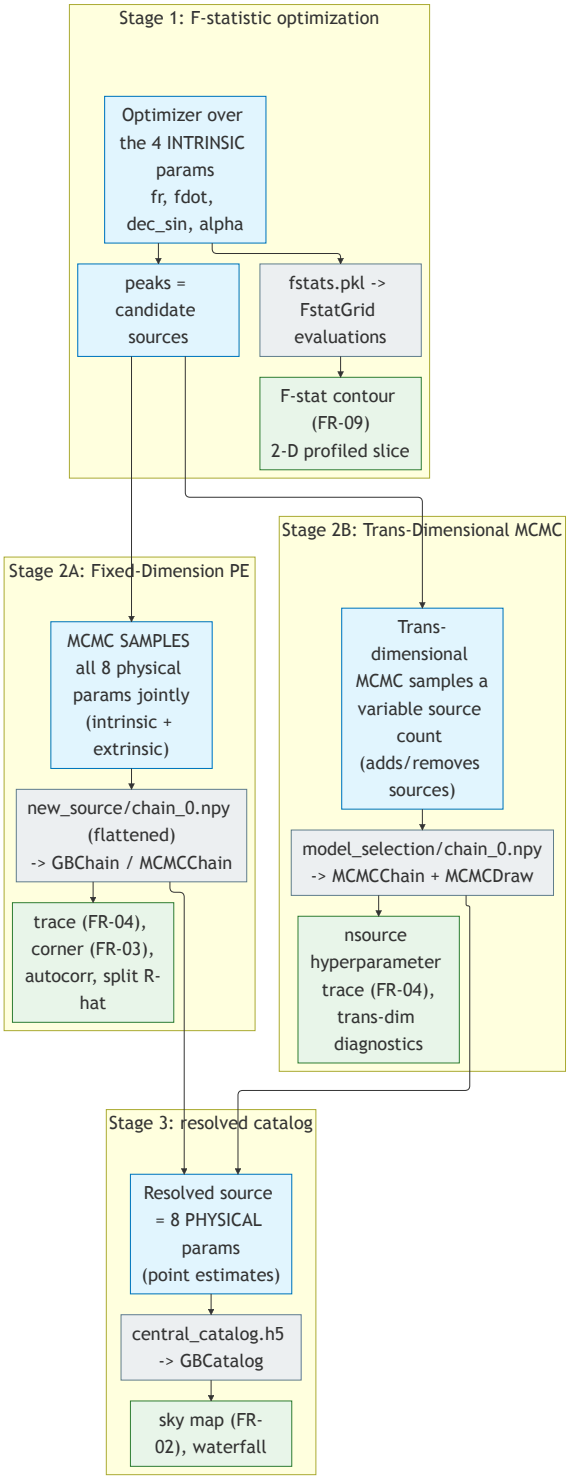

Figure 2.3: **Parameter flow of the Galactic Binary pipeline.** The F-statistic grid scores the 4 intrinsic parameters while the 4 extrinsic ones are recovered analytically by the least squares fit. The parameter estimation MCMC then samples all 8 jointly, so the resolved catalog carries the full 8 as point estimates.

Table 2.1: **Physical parameters describing a Galactic Binary source.** Parameters 1–2 and 4–5 are intrinsic. Parameters 3 and 6–8 are extrinsic.

#	Parameter	Symbol	Units / Description
1	Frequency	f_0	GW frequency at reference time [Hz]
2	Frequency Derivative	$\dot{f}$	Rate of orbital decay due to GW emission [Hz/s]
3	Amplitude	$\mathscr{A}$	Dimensionless strain amplitude
4	Declination	δ	Sky position in equatorial coords [rad]
5	Right Ascension	α	Sky position in equatorial coords [rad]
6	Inclination	ι	Angle between orbital angular momentum and line of sight [rad]
7	Polarization	ψ	Orientation of orbit in the sky plane [rad]
8	Initial Phase	ϕ_0	GW phase at $t = 0$ [rad]

2.1.4 The User Experience Gap

While the GlobalFit Framework provides the necessary abstraction for orchestrating these distributed modules, Basset (2024) identifies a clear gap in the user experience: once the pipeline runs, how do scientists actually make sense of its output?

Because thousands of sources are processed in parallel, standard debugging techniques simply do not apply. Without dedicated visual tools, the system remains a black box. Basset calls for two distinct visualization modalities:

- **Operation Dashboards:** For operators to handle module scheduling, resource adaptation and runtime debugging without wasting computational resources.
- **Monitoring Dashboards:** For science experts to actively monitor the progress of estimators and view a live synthesis of the source catalog.

Making sense of that output therefore needs interactive visual frameworks and monitoring dashboards that support the scientific debugging and "steering" of the Global Fit.

2.2 Bayesian Inference and MCMC Foundations

Visualization design depends on the statistical machinery that produces the data the dashboard must display, so that machinery is worth reviewing first. The Global Fit is, at its core, a Bayesian inference problem and the outputs that need to be visualized are MCMC chains. This section introduces the concepts and tools that directly inform the visualization requirements.

2.2.1 Markov Chain Monte Carlo Methods

Bayesian inference seeks to compute the posterior distribution of the model parameters θ given the observed data d:

$$P(\theta \mid d) = \frac{P(d \mid \theta)\,P(\theta)}{P(d)} \tag{2.1}$$

where $P(d \mid \theta)$ is the likelihood, $P(\theta)$ is the prior and $P(d)$ is the evidence (a normalization constant). For complex models with many parameters, such as the eight-dimensional GB waveform model or the full Global Fit with thousands of simultaneous sources, this integral is analytically intractable, which means it cannot be solved in useful time.

MCMC methods solve this problem by constructing a Markov chain whose stationary distribution is the target posterior (Gelman et al. 2013). After running the chain for enough steps, the samples it generates are effectively drawn from $P(\theta \mid d)$. They can then be used to estimate means, credible intervals and correlations.

That chain is built one step at a time by the Metropolis-Hastings sampler (Metropolis et al. 1953; Hastings 1970), which proposes a new point at each step and accepts or rejects it. Whether the resulting path settles and how thoroughly it explores is what tells the physicist whether the samples can be trusted as draws from the posterior.

Several diagnostic concepts arise directly from MCMC theory and dictate the core set of visualizations the library must support. Because the Global Fit posterior is defined over thousands of dimensions, the physicist cannot inspect convergence analytically. Instead, each of the following diagnostics gives a specific geometric view of how the walkers behave as they explore the parameter space.

A trace plot shows one parameter against the iteration index, where a converged chain settles into a flat, well mixed band (Figure 2.4). The drifting opening before that band is the burn-in, discarded before any statistic is computed. Because sub-bands converge at different rates, the burn-in is assessed for each on its own.

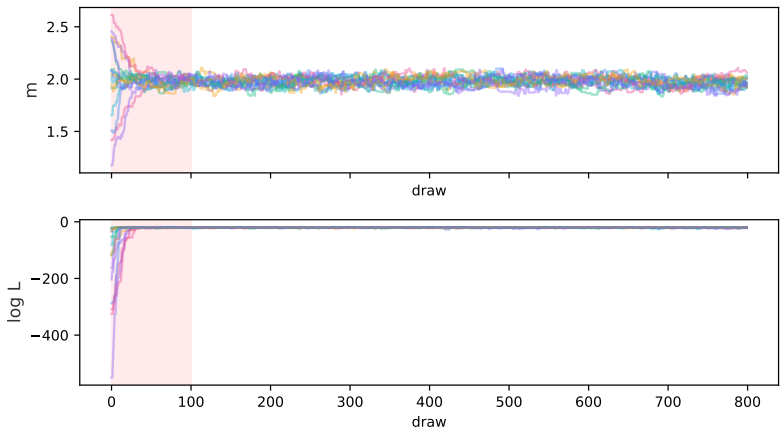

Figure 2.4: **Trace plot of a single parameter across an MCMC run.** An early drift from the starting value gives way to a flat, well mixed band, the signature of a stationary chain after the burn-in is discarded.

The autocorrelation function reports the autocorrelation time τ, the number of steps between two effectively independent samples (Figure 2.5). A chain of length N then holds only about N/τ independent draws, so a slow fall toward 0 means the run must be extended. The acceptance fraction, healthy between about 0.2 and 0.5, is a cheaper second check.

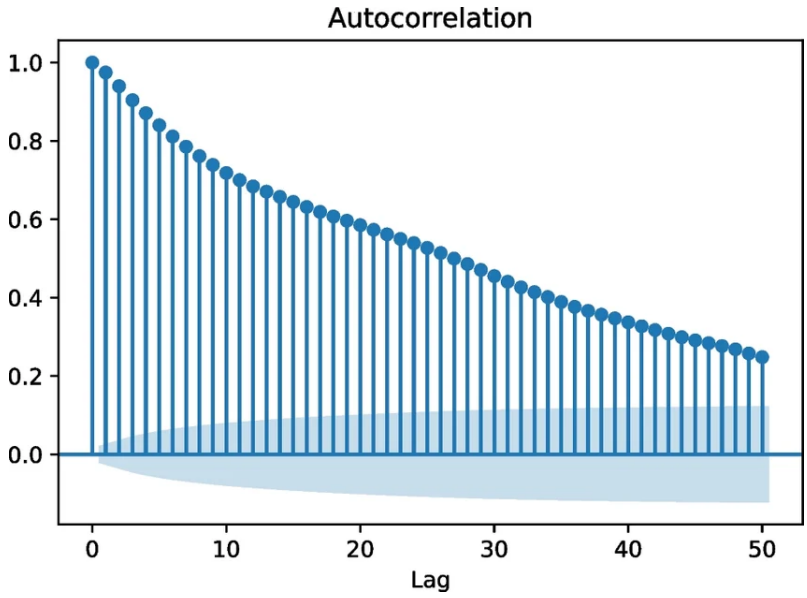

Figure 2.5: **Autocorrelation function.** The correlation of a sampled series with itself against the lag, falling from 1 toward 0 as the lag grows. A slower fall means a larger autocorrelation time τ and fewer independent samples. *Source: (Shi et al. 2023).*

A corner plot summarizes a multi parameter posterior as a triangular grid: 1D marginals on the diagonal and pairwise joint distributions as nested contours below (Figure 2.6). A contour stretched into a tilted ellipse marks a degeneracy, a pair the data constrains only in combination.

2.2.2 The emcee Ensemble Sampler

A widely used MCMC sampler in astrophysics is `emcee` (Foreman-Mackey et al. 2013), built on the ensemble method of Goodman and Weare (2010). Plain Metropolis-Hastings runs a single walker that proposes a step and then accepts or rejects it, the picture from earlier. `emcee` instead runs many walkers together and couples them, so each walker aims its next step using where the other walkers currently sit, not as a fixed blind jump. Because the proposals follow the spread of the ensemble, the walkers settle along whatever shape the posterior has, even where correlated parameters stretch it into a narrow diagonal, which spares the user from tuning the step size by hand.

2.2.3 Trans Dimensional Sampling

The Global Fit introduces a challenge that standard MCMC cannot handle: the *number* of sources is itself unknown. In a conventional MCMC run, the dimensionality of θ is fixed. But in the LISA data stream, the pipeline must simultaneously determine how many GB sources are present in each frequency band and what their parameters are. This is a trans-dimensional inference problem.

Two samplers handle it in the Global Fit, by different routes. **Eryn** uses Reversible Jump MCMC (Green 1995): built on `emcee`, it adds "birth" and "death" moves that insert or remove a source on top of the usual ensemble updates, so the dimension grows and shrinks as the run

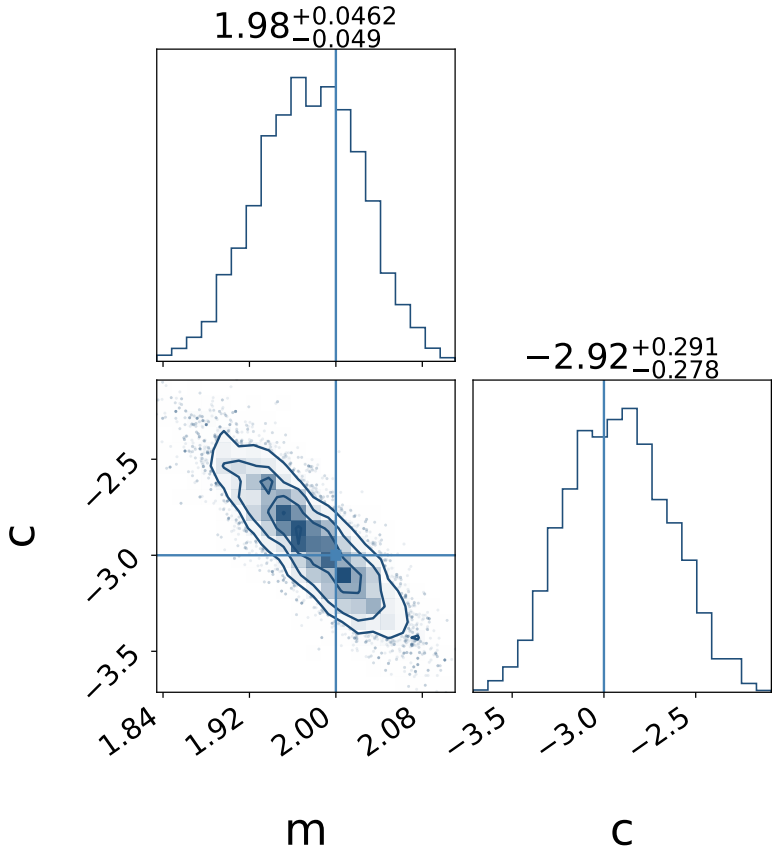

Figure 2.6: **Corner plot of a multi parameter posterior.** The diagonal holds the 1D marginal of each parameter and the lower triangle the pairwise joint posteriors as nested contours, where a tilted contour marks a degenerate pair. Generated with `corner.py` (Foreman-Mackey 2016).

proceeds (Karnesis et al. 2023). **Jexplore**, a JAX-based sampler, uses a product space approach (Carlin and Chib 1995): it fixes a generous upper bound on the sources a band can hold and keeps that many slots in the state throughout, switching each slot on or off so the active count varies while the dimension stays fixed.

For visualization the route matters less than its outcome. Both samplers hand the pipeline the same fact, a per draw list of sources whose length varies, so this work models that list directly rather than the mechanism behind it. The views stay valid whichever sampler produced the chains, including one the pipeline has not adopted yet.

2.3 Dashboard and Visualization Fundamentals

The literature separates general purpose user interfaces from data intensive scientific dashboards, which differ in taxonomy, visual perception and interaction models.

2.3.1 Functional Taxonomy and Design Mining

Historically, dashboards were defined as "at a glance" monitoring displays (Few 2006). Sarikaya et al. (2019) looked at how dashboards are actually used and found they do not all serve the same purpose: some are built to be read at a glance, others for digging into the data and working through an analysis.

Turning this distinction into a concrete design means grounding it in interfaces that already work. Lin et al. (2024) propose "design mining", studying hundreds of existing dashboards to extract the visual patterns that recur. Parsons (2022) makes a related point: designers rely on precedent and inspiration. The concrete output is a catalog to draw on: the 42 dashboard design patterns of Bach et al. (2023), distilled from a review of 144 real-world dashboards and covering both their content and their layout. The principle guiding their selection is that each visual component should serve a specific analytical goal (Sedrakyan, Mannens, and Verbert 2019).

2.3.2 Visual Perception and Cognitive Load

The efficacy of a scientific dashboard is constrained by human cognitive limits and should be adapted to them. Islam and Jin (2019) note that the brain reads graphical patterns far faster than the same values in a table, but at the scale of the Global Fit the point is sharper than speed: a catalog of millions of overlapping sources cannot be read as numbers at all, so a visual encoding is the only access to its structure rather than a quicker one. Few (2006) argues that effective design must prioritize “preattentive processing”: the eye perceives attributes like color and orientation in under 200 milliseconds, before conscious attention engages, so a high-contrast color reserved exclusively for anomalies or critical data points makes the exception visible before it is searched for. That property only holds if no decorative color occupies the same channel, which is the discipline Few imposes: every color must encode a quantity.

These perceptual limits are the reason scientific figures need a different discipline from commercial graphics. Midway (2020) makes the point directly: figure making is rarely taught formally in science, so researchers often accept whatever a plotting tool produces by default. Writing for researchers rather than for business intelligence audiences, he sets out ten principles for effective scientific visualization, the core being that every graphical choice encodes a quantity rather than decorating, that color in particular always carries information, that uncertainty is shown rather than hidden behind point estimates, that data stays visually distinct from models.

That priority is what separates a scientific visual standard from the engagement driven aesthetics of many commercial dashboards: visual embellishment that does not encode a quantity competes for attention, so a scientific display leaves it out.

Recent eye-tracking research clarifies that while behavioral reaction times vary minimally across layouts, physiological data reveals hidden differences in cognitive strain. Zhang et al. (2024) demonstrate that positioning an interface's "Core Chart" in the left-central zone optimizes visual search compared to traditional mid-central symmetry. As shown in the hotspot diagnostics in Figure 2.7, a centered layout forces repetitive, intensive viewing across secondary windows (dense red zones), while a left-central anchor follows natural left-to-right reading habits, showing more efficient green scanning paths and lower red zones that translate to lower cognitive load.

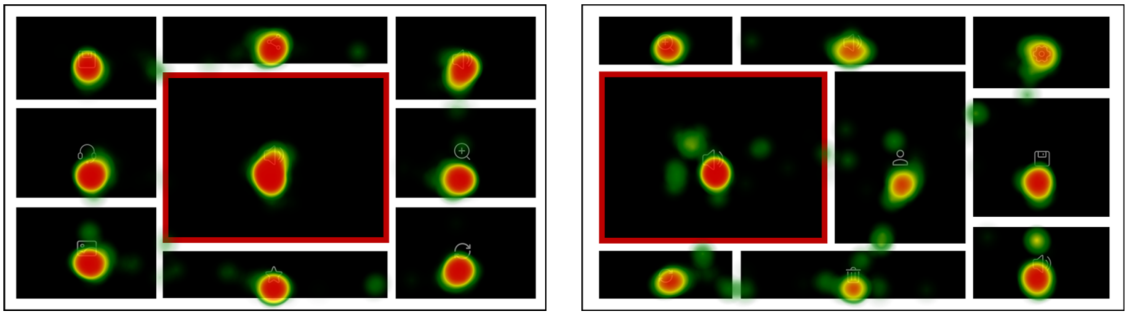

Figure 2.7: **Hotspot maps comparing cognitive load by layout order.** The traditional mid-central layout (left) requires repetitive gazing (red zones). The better, left-central layout (right) shows improved scanning (more green zones and less red zones) by aligning with natural reading paths. *Source: (Zhang et al. 2024).*

Few (2006) also establishes that the top-left position is the most explored on a dashboard, so the chart placed there must be chosen for its relevance to the primary task. The DMiner study of Lin et al. (2024) grounds this in Shneiderman's layout hierarchy: the overview, carrying fewer fields and coarser information, belongs at the top or left as the entry point, while detailed multi-field views sit at the bottom or right. The same study formalizes Tufte's "small multiples" as an expert layout rule: views of the same visual type should be arranged together at the same width, height and scale to facilitate direct comparison rather than relying on working memory (Lin et al. 2024).

To separate these semantic zones without adding visual clutter, Few (2006) advocates relying on negative space rather than heavy gridlines or boxes, while keeping the core chart and its immediate supporting views tightly grouped within a single eye span. Rigid layouts suit common users, but

Vuckovic, Wolosiuk, and Schmidt (2024) show that domain experts need adaptable environments where they can reposition panels to fit the tool to their evolving analytical workflow.

In the high density environment of LISA telemetry, color discriminability also becomes an important factor. Szafir (2018) established that the ability to distinguish colors decreases as the size of the visual mark decreases. Countering this requires perceptually uniform colormaps, where an equal change in the encoded value corresponds to an equal perceptual change in color, avoiding the false edges of traditional rainbow scales. Szafir's modeling techniques can then space these colors further apart to keep them distinct even for "fine grained" data elements. Wang et al. (2020) support visualization literacy through modular cheat sheets of six kinds: anatomy, construction, visual patterns, pitfalls, well known relatives and false friends. For a general audience all six help, but the expert audience here changes which are worth including. The physicists already read a corner or trace plot fluently, so anatomy and construction sheets would only restate what they know. The pitfalls and false friends modules are the ones that carry weight, because the risk for an expert is not failing to read a chart type but misreading a feature that the tool's own rendering introduced.

2.3.3 Interaction Models and Cooperative Design

Interactivity is what transforms a static data display into a tool that scientists can actually use to explore their data. For iterative analysis like the Global Fit, Yi et al. (2007) argue that interaction must go beyond simple navigation to support intent driven tasks such as filtering noise or reconfiguring parameters. To connect the physics domain to the interface design, the architecture applies the task typology from (Brehmer and Munzner 2013). The typology treats the iterative global fit steps as a chain of interdependent operations, where the output of a noise estimation task becomes the input for binary source identification, so the interface can move fluidly between high level monitoring and detailed data inspection.

Interpreting the output of stochastic pipelines also raises questions of trust and comprehension, especially when the developer of the visualization is not a domain expert in the field where conclusions are drawn. Setlur et al. (2024) identify a frequent "breakdown in grounding" when users interact with opaque algorithms and propose that dashboard design should follow a "Cooperative Principle" where the system helps the user resolve ambiguity through context and explanation.

This aligns with the findings of Alhamadi (2020), whose survey of dashboard challenges and adaptations frames the problem as a tradeoff: placing too much data in a single view causes information overload and misleading interpretation. The remedy is an adaptation technique rather than more data. Bach et al. (2023) classify expert systems as "Analytic Dashboards" noting that they must avoid overflow pagination since vertical scrolling breaks the visual continuity needed to compare charts. Instead, they mandate a "Screenfit" design that uses filters, tabs and details-on-demand to manage dataset scale within a single view. For the high dimensional Global Fit output a static view is not enough, so the interface lets the physicist adjust granularity by drilling down from the population overview to a single source, which is how a result is checked for physical validity rather than accepted whole. This drill-down relies on Coordinated Multiple Views (Bach

et al. 2023), the "coordinated when related" pattern mined by Lin et al. (2024), where selecting a source in the overview automatically drives the detailed supporting views.

Trust in an opaque algorithm is earned rather than assumed. McKinley, Pandey, and Ottley (2025) establish that this requires "Trustworthy Design" through familiarity and provenance: the interface must explicitly disclose data sources and utilize standard, recognized chart types rather than novel, unfamiliar graphics that increase skepticism.

2.3.4 Managing High-Density Data

Visualizing heavy, high dimensional datasets runs into a problem called "overdraw", where points pile up so densely that they obscure the underlying distribution. For the "Sangria" dataset, which contains millions of overlapping sources, standard scatterplots will fail to convey the necessary structural detail. To address this, Mayorga and Gleicher (2013) present "Splatterplots", a hybrid technique that prioritizes the preservation of outliers while abstracting dense regions into smooth contours. For astrophysical analysis, this distinction matters: researchers perceive the global structure of stochastic foregrounds (confusion noise) without losing the individual signals that often appear as visual outliers (Figure 2.8).

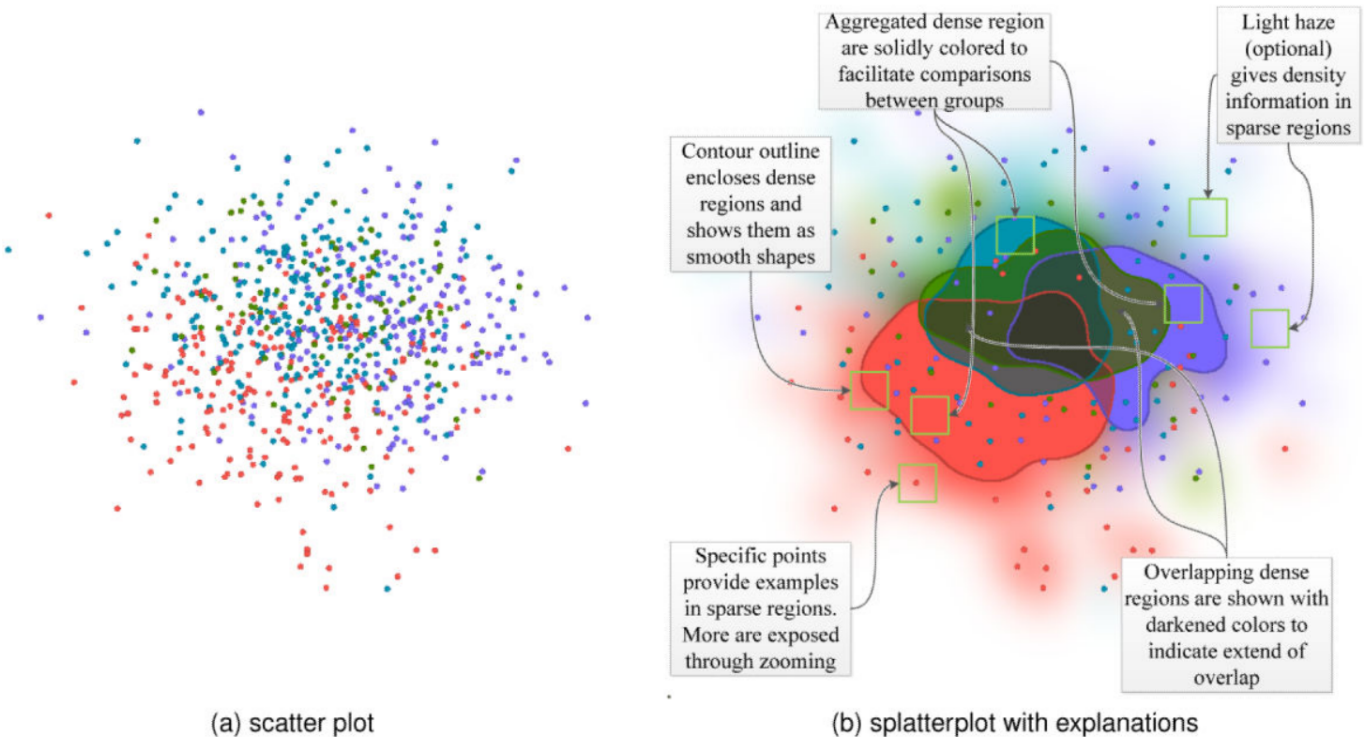

Figure 2.8: **Overcoming overdraw with Splatterplots.** Standard scatterplots (left) obscure density in high-volume datasets like Sangria. Splatterplots (right) abstract dense regions into contours while preserving outlier visibility. *Adapted from Mayorga and Gleicher (2013).*

2.4 Existing Plotting Limitations in LISA Global Fit

The theoretical literature provides the principles, but the specific architectural requirements of the Global Fit visualization library become clearest when looking at what already exists and where it falls short. The current codebase relies on ad hoc scripts written by physicists to validate specific mathematical outputs, rather than on any organized software engineering architecture.

2.4.1 Native Pipeline and Sampler Plotting Limitations

While the underlying samplers are mathematically rigorous, their native visualization is insufficient for the analytical needs of the LISA mission. The pipeline provides basic diagnostics through ad hoc scripts (e.g., `diagnostic.py`) and generic sampler methods, which share several major flaws:

- **High Coupling and Fragility:** The plotting functions are often tightly coupled to the specific `pandas` schemas of individual pipeline runs. If the HDF5 layout changes or a new parameter is introduced, the plotting scripts often break, requiring manual intervention.
- **Lack of Interactivity:** The outputs are primarily static images (e.g., standard `matplotlib` PNGs or PDFs). As discussed in Section 2.3, static views are insufficient for high dimensional datasets like the "Sangria" catalog, where scientists need to interactively "drill down" to verify the physical validity of automated results (Alhamadi 2020).
- **Scalability Issues:** The native scripts provide no view that scales to the full Galactic Binary population, which reaches tens of millions of sources at mission scale. A plain `matplotlib` scatter at that scale would saturate under overdraw, so reading the population needs the density reduction the native diagnostics do not provide.
- **Representationally Limited:** Native outputs are raw parameter traces and static 1D/2D corner plots, fine for a basic convergence check. None of them represents the trans-dimensional structure of the posterior, where the number of sources varies from draw to draw, so that variation has no native view at all.
- **No Reusable Interface:** The native scripts produce their figures and stop there, exposing no domain model or stable Application Programming Interface (API) that a larger interactive tool could build on. Reusing them would mean rewriting the data access from scratch.

2.4.2 ArviZ and the Limits of General Purpose Visualization Tools

A prominent general purpose tool for Bayesian model diagnostics is ArviZ (Martin et al. 2026), a Python library that provides a unified interface for generating trace plots, corner plots and posterior predictive checks from a variety of samplers. Any discussion of a new visualization library for MCMC data must address why ArviZ is insufficient for the problem at hand, rather than simply ignoring its existence.

The limitations fall into two categories. The first is a domain limitation. While the modern modularization of ArviZ 1.0 introduced the `arviz-plots` subpackage and `PlotCollection` class to support custom visualization development, it remains a domain agnostic statistical framework. It operates on generic named dimensions (chain, draw, parameter name) and has no concept of what a Galactic Binary is, what ecliptic coordinates mean or how to project source parameters onto a spherical sky map. It cannot render a Power Spectral Density residual comparison against the LISA sensitivity curve because it has no awareness of gravitational wave physics. The domain specific diagnostics that the LISA Global Fit requires simply do not exist natively in ArviZ. They must be constructed by using ArviZ's new modular features as an underlying computation engine to power a domain specific frontend.

The second limitation is one of scope rather than capability. ArviZ is a diagnostics and generic MCMC plotting library, not an ingestion layer: it operates on clean aligned arrays that a caller has already assembled. The Global Fit hands over no such arrays. It writes its output across several on-disk formats that change as the pipeline evolves, structured HDF5 backends, NumPy chain arrays and pickled grids, which have to be read and assembled into a coherent model before any diagnostic can run. ArviZ does none of that reading. What ArviZ does supply well is the diagnostic mathematics. The modularized `arviz-stats` package exposes a lightweight array interface that depends only on NumPy and SciPy, an interface "intended for experienced users and library developers" (Martin et al. 2026) that computes R-hat, the effective sample size and the autocorrelation time directly from arrays without the `xarray` container. A domain specific frontend can therefore consume `arviz-stats` for the convergence mathematics while doing the ingestion, the domain modeling and the astrophysical visualization that ArviZ was never meant to provide.

The same reasoning extends beyond scientific libraries to the commercial business intelligence platforms that might be proposed in their place, such as Tableau or Power BI. Two properties rule these out before any feature comparison. The first is the data model: they ingest rectangular relational tables through database or spreadsheet connectors and do not chunk load multi gigabyte HDF5 arrays through partial reads, so the streaming ingestion the Global Fit needs has no place in them. The second is distribution: they are proprietary and licensed, whereas the open science setting of the LISA consortium, modelled on the open agency dashboards discussed in Chapter 1 (Anghelea et al. 2023), calls for a tool every member can install and run without a commercial entitlement. Toasa et al. (2018) make the general form of this argument from the dashboard side: an accurate and understandable display of specialized information often requires customizing existing platforms or building specific boards rather than accepting a generic one. For the Global Fit the customization required runs deep, reaching the data model, the sky projection and the confusion aware density abstraction, so a dedicated library is the lighter path rather than the heavier one.

2.5 Synthesis and Research Objectives

The review presented in this chapter shows that the Global Fit poses several interrelated challenges. One of the most pressing is the Human-Computer Interaction problem: the volume of overlapping signals makes standard static outputs difficult to interpret without a dedicated visualization layer.

The existing ad hoc scripts (`diagnostic.py`) and native sampler plotting tools do not address the unique visualization demands of the Sangria dataset. They provide only basic diagnostic outputs, lack the abstraction needed to handle high density probabilistic data interactively and are not built with the software engineering rigor needed to eventually power a centralized mission dashboard.

The gap can be summarized in two points. First, there are no visualization tools specifically tailored to the confusion noise and trans-dimensional nature of gravitational wave telemetry. Second, there is no decoupled software engineering architecture capable of translating raw HDF5 MCMC outputs into interactive visual models. This dissertation addresses both.

Chapter 3

Research Methodology

This chapter defines the research methodology, requirements engineering process and evaluation framework that govern the development of the LISA Global Fit visualization library. The work follows a Design Science Research (DSR) framework with the goal of producing a well engineered IT artifact that addresses the specific challenges of visualizing high density, trans-dimensional MCMC data. The system architecture that results from this methodology is presented in Chapter 4.

3.1 Design Science Research (DSR) Framework

This thesis adopts Design Science Research (DSR) as its primary methodology. Applying a formal methodology distinguishes a scientific thesis from a standard software engineering project. Standard projects satisfy immediate client requirements. DSR instead produces a novel IT artifact to solve a generalized class of problems and generates prescriptive, mid range design knowledge (such as design principles or architecture patterns, formalized as λ-knowledge) for reuse by the broader academic community (Brocke, Hevner, and Maedche 2020).

In this thesis the two DSR outputs are distinct. The artifact is the visualization library and its layered architecture. The design knowledge is the reusable account of how to map trans-dimensional gravitational wave MCMC output, where the number of resolved sources changes from draw to draw, onto a decoupled domain model and a density reduced visual layer, while reusing an existing diagnostics engine for the convergence statistics rather than rebuilding them. The architecture and the design knowledge behind it are meant to transfer to other trans-dimensional inference problems and to the further source types the same mission will add, while the layout is one instantiation of it.

3.1.1 Methodological Justification: Why DSR?

Several alternative methodologies were evaluated and rejected before selecting DSR. Action Research prioritizes the sociotechnical process of change within a specific organization (Baskerville 1999), which Iivari and Venable frame as a behavioral science concern with truth rather than the design science concern with the utility of a constructed artifact (Iivari and Venable 2009), whereas this thesis seeks a reusable, technology agnostic architecture whose value lies in whether

it works and transfers rather than in an account of how the consortium works. The Nunamaker systems development paradigm (Nunamaker and Chen 1990) is a genuine candidate, since this thesis produces a software package, but it makes the running system the primary contribution, whereas the contribution claimed here is the reusable architecture behind the dashboard rather than the dashboard itself. Case studies and surveys are observational, investigating a phenomenon without controlling it (Yin 2018; Hevner et al. 2004), whereas this work intervenes by building a novel artifact intended to change how astrophysicists inspect trans-dimensional MCMC data. DSR is selected because it is the only evaluated framework that treats the creation of the artifact itself as scientific research while providing a structured cycle to contribute reusable design knowledge back to the academic community.

3.1.2 The DSRM Process Model

To structure the execution of this research, this thesis adopts the Design Science Research Methodology (DSRM) process model proposed by Peffers et al. (2007). The primary objective of DSRM is to provide a shared mental model for design science research outputs, distinguishing them from traditional theory testing or interpretive research.

The process model consists of six activities in a nominal sequence as illustrated in Figure 3.1. Depending on the origin and nature of the research, the DSRM supports multiple possible entry points. These include problem centered, objective centered, design centered and client initiated paths.

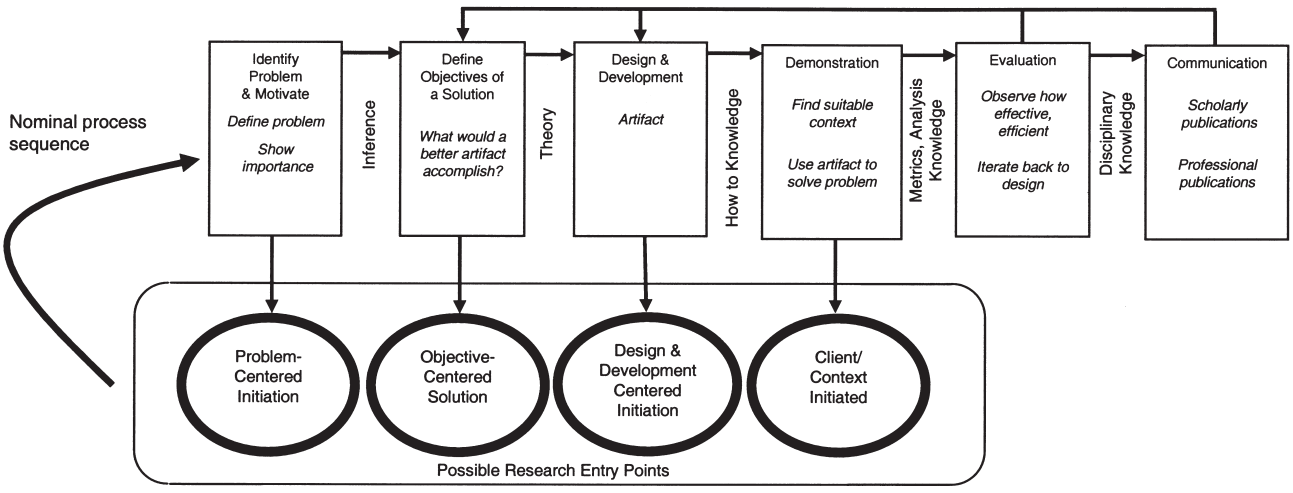

Figure 3.1: **DSRM Process Model.** The six activities of the Design Science Research Methodology in a nominal sequence showing the possible research entry points (Peffers et al. 2007).

This thesis follows a problem centered entry point starting at Activity 1. The practical limitations observed within the LISA Global Fit pipeline trigger this choice. Specifically, astrophysicists face difficulties parsing, diagnosing and visualizing trans-dimensional MCMC outputs.

Peffers et al. (2007) note that this nominal sequence can also order the presentation of the research. The structure of this document follows it. Problem Identification and Motivation (Activity 1) is the work of Chapter 1 and the gap analysis in Chapter 2. The objectives of a solution (Activity 2) are defined in this chapter as the requirements of Section 3.2. Design and Development (Activity 3) is split across two chapters: Chapter 4 presents the architecture and Chapter 5 realizes it. Demonstration (Activity 4) is the application of the library to Global Fit run data in Chapter 5. Evaluation (Activity 5) is Chapter 6. Communication (Activity 6) is the thesis itself, summarized in Chapter 7. The chapter order is therefore not arbitrary but the DSRM sequence applied to the document.

3.1.3 Methodological Integration

The four frameworks used in this thesis operate at distinct levels, each with a specific role. Together they form one research design rather than a set of independent choices.

DSR acts as the macro level research frame. It establishes software construction as valid scientific inquiry and requires the final artifact to be demonstrated and evaluated against defined objectives (Hevner et al. 2004). All other frameworks operate within this structure.

Within the DSR cycle, NASA Systems Engineering provides the process to translate vague scientific needs into strict and traceable requirements (NASA 2016). Without this translation, the DSR evaluation phase (Activity 5) lacks objective criteria for testing. The two frameworks couple directly: NASA SE defines the success conditions and DSR provides the formal research cycle to evaluate them.

Because the artifact is a scientific visualization system, the data driven visualization design process (Walny et al. 2020) is embedded within the DSR development phase. It governs how raw HDF5 data maps to visual representations, how overdraw is handled in high density posterior plots and how visual encodings are designed against the underlying physics. This is a domain specific requirement that neither DSR nor NASA SE address independently.

Finally, the ECSS software engineering standards (European Cooperation for Space Standardization 2025), adopted here as the project's engineering baseline, bound the implementation. While the standard does not explicitly prescribe specific design paradigms, it calls for verifiable internal consistency, explicit interface dependencies and strict requirements traceability at the code level. The low coupling and strict encapsulation of the proposed architecture are the specific design strategies chosen to satisfy that discipline. The ECSS verification process also feeds back into DSR Activity 5 to provide the formal evidence required to evaluate the artifact.

Together these four frameworks divide the work: DSR structures the research, NASA SE defines what to build, the visualization design process dictates how to render the data and ECSS governs how to write the code.

3.2 Requirements Engineering and User Perspective

This section defines the objective criteria required to evaluate the final visualization library. Under the Design Science Research (DSR) framework, Activity 2 dictates that a solution must have defined objectives to allow for formal evaluation. However, DSR does not prescribe a specific method to capture these objectives. This project solves this by applying the NASA Systems Engineering (NASA SE) technical requirements definition process. NASA SE provides the mechanism to translate informal stakeholder expectations into formal engineering constraints (NASA 2016).

Astrophysicists describe their needs through workflows rather than software specifications. Following NASA SE guidelines, this project first captures these informal operational needs as user scenarios. These scenarios are then systematically decomposed into technical requirements using definitive "shall" statements. Finally, following the ECSS software engineering standards (European Cooperation for Space Standardization 2025), the project maintains strict bidirectional traceability. Every formal requirement maps directly back to a specific user scenario. With this traceability, the final software evaluation (DSR Activity 5) tests exactly what the LISA physics working group requested.

The requirements presented here derive from three sources: the analytical needs identified by the LISA physics working group (Basset 2024), the software engineering gaps identified in Chapter 2 and direct consultation with the astrophysicists operating the Global Fit pipeline.

These sources share one defining property: the audience is uniformly expert. The scenarios that follow are those of practising physicists and pipeline developers, not of a general or citizen science public. Vuckovic, Wolosiuk, and Schmidt (2024) show that this distinction changes the design itself and not the wording alone, because the needs of practitioners and those of lay audiences diverge. A lay audience needs quantities abstracted into simplified indicators, while practitioners need the underlying measures kept intact so they can judge them directly. Designing for the expert end has three concrete consequences here. The interface uses the parameter names the physicists already use rather than an explanatory layer that would slow them down (Section 4.3). It keeps the raw diagnostics in reach, the MCMC trace plots, the sampled parameter posteriors, the F-statistic contours and the Power Spectral Density (PSD) residuals, rather than replacing them with simplified summaries, so that the automated convergence flag (FR-05) is a starting point an expert can check against the chains themselves rather than a verdict they must accept. It also assumes statistical and physical literacy, so a corner plot or a confusion limited spectrum can be shown without tutorialization. This does not license clutter. The expert audience justifies retaining analytical substance, but it does not relax the perceptual constraints set out elsewhere in this work: density reduction, color discriminability and the core chart layout still apply, because an expert reading a dense display is subject to the same cognitive limits as anyone else.

3.2.1 User Scenarios

US-01: A LISA physicist finishes a Global Fit run on the CNES Trex cluster. She needs to review the recovered Galactic Binaries, their sky distribution and the residual noise. The raw HDF5 dataset

is 1.3 TB, making a full transfer impractical. Instead, she downloads only the lightweight summary catalogs (e.g., `central_catalog.h5`) to her laptop. She needs the visualization library to read these summary files locally and plot them interactively without running a standard laptop out of memory.

US-02: A researcher notices that the recovered source count dropped between iteration 3 and iteration 4. He needs to isolate and inspect the MCMC trace plots for specific frequency subbands to check whether any chains failed to converge or got stuck in local minima.

US-03: During a collaboration meeting, a physicist needs to show a corner plot of a specific Galactic Binary's posterior to demonstrate that its frequency and sky position are well-constrained. She needs to generate a publication quality figure directly from the library.

US-04: A pipeline developer wants to verify that the noise model is being subtracted correctly. He loads the PSD residual plot to compare the raw data spectrum against the reconstructed signal plus noise model, looking for excess power that would indicate a missed source.

US-05: A pipeline developer is testing a new frequency band and needs to understand how strongly the source parameters trade off against each other before launching a computationally expensive MCMC run.

3.2.2 Functional Requirements

Table 3.1 lists the functional requirements derived from the user scenarios. This formal mapping is required by ECSS standards for traceability. For example, the informal need to find a chain that did not fit (US-02) is decomposed into strict engineering objectives (FR-04 and FR-05) that can be objectively tested during DSR Activity 5. The requirement for interactive sky maps with density reduction (FR-02) is strictly justified by the scale of the new Common Dataset 1 Light (Mojito Light), which injects 15,539,324 Galactic Binaries (LISA DDPC 2026a). Plotting that many sources with standard scatter plots would produce catastrophic visual overdraw.

The LDC2a "Sangria" dataset (Cécile Cavet 2022) sharpens a different requirement. Sangria is confusion limited: roughly 30 million Galactic Binaries overlap with Massive Black Hole Binary signals and instrumental noise, so most sources are not separable in any single static view. Isolating a resolvable source from this stochastic foreground requires inspecting it across the time, frequency and parameter domains rather than in one fixed projection. This is the justification for filtering the catalog and the chains by frequency subband (FR-08) and for the frequency domain reconstruction views described in Chapter 5, which let the physicist narrow to one band and examine the overlapping tracks a region at a time.

ID	Requirement	Source
FR-01	The library shall parse Global Fit run directories, across their several on-disk formats, into structured Python domain models.	US-01
FR-02	The library shall render interactive sky maps showing the spatial distribution of resolved Galactic Binaries.	US-01
FR-03	The library shall render N-dimensional corner plots from MCMC posterior chains.	US-03
FR-04	The library shall render trace plots showing individual walker trajectories, including the `nsource` hyperparameter to visualize trans-dimensional model jumps.	US-02
FR-05	The library shall compute a convergence heuristic from the raw MCMC chain data and flag candidate chains for possible non-convergence.	US-02
FR-06	The library shall render Power Spectral Density (PSD) residual plots comparing observed data against reconstructed models.	US-04
FR-07	The library shall support exporting generated plots as publication quality static images (PNG, PDF).	US-03
FR-08	The library shall allow users to filter and isolate the GBCatalog and MCMC trace visualizations by specific frequency subbands.	US-02
FR-09	The library shall render 2D contour surfaces of the F-statistic grid evaluations to visualize parameter degeneracies before full MCMC sampling.	US-05

Table 3.1: Functional Requirements for the Visualization Library.

3.2.3 Non-Functional Requirements

Table 3.2 lists the non functional requirements that constrain the architecture and implementation. These constraints are not stylistic preferences. Because LISA is an ESA mission, the software in its ground segment is developed under the ECSS software engineering standard ECSS-E-ST-40C (European Cooperation for Space Standardization 2025), which requires specified requirements, a maintainable and verifiable design and traceable verification across the development lifecycle. This project adopts that standard as its engineering baseline, because a library meant for that segment should be built the way the segment builds software rather than be re-engineered to the standard after adoption. That choice is the reason a quick plotting script is not an acceptable deliverable: the standard asks for a hierarchical architectural decomposition, explicit interfaces and requirements traceability at the level of the code itself. Low coupling and strict encapsulation are the architectural

strategies adopted here to fulfill those demands. Each non functional requirement below traces to that goal. The architecture presented in Chapter 4 exists in large part to satisfy it.

ID	Requirement	Rationale
NFR-01	The library shall maintain strict separation between data ingestion, domain models and rendering modules (low coupling, high cohesion).	Maintainability, ECSS-E-ST-40C
NFR-02	The library shall strictly bound RAM usage during rendering to prevent out-of-memory crashes, given that uncompressed L2-to-L3 pipeline outputs reach 20 GB (LISA DDPC 2026b) and raw Mojito datasets exceed 1.3 TB (LISA DDPC 2026a).	Performance Efficiency (US-01)
NFR-03	The library shall be installable as a standard Python package via pip, with no dependencies on specific HPC environments.	Flexibility / Installability (US-01)
NFR-04	The library's Data Ingestion Layer shall implement an abstract interface for catalog loading, so that the system stays expandable to future source types (e.g., MBHBs, EMRIs, Noise) and breaking changes in upstream output formats (e.g., L2-to-L3 batching and `snake_case` schema updates) can be accommodated without modifying the core visualization domain.	Flexibility / Adaptability
NFR-05	The library's Data Ingestion Layer shall load run data on demand, reading only the catalog blocks, chain segments or draws a view requests rather than the whole run directory, so that inspection of a run can start on a standard consumer laptop without ingesting the full dataset first.	Performance Efficiency (US-01)

Table 3.2: Non-Functional Requirements for the Visualization Library.

NFR-02 and NFR-05 look similar because both answer the data volumes above, but they forbid different failures and US-01 contains both. NFR-02 bounds the peak, so a population view cannot exhaust memory however large the catalog grows. NFR-05 bounds the entry cost, so inspecting one band or one source never requires ingesting the whole run first. Either can hold without the other. The object materializing read of Chapter 6 makes the first case concrete, because it loads exactly what it is asked for yet its cost grows with the source count, which is why the population views render through the streaming reduction instead. A design that converts the whole run up front shows the second, since it can render within bounds afterwards yet makes the physicist pay for the

full dataset before the first plot. Under ISO 25010 the two serve Performance Efficiency through different subcharacteristics, resource utilization for NFR-02 and time behaviour for NFR-05.

3.3 Evaluation Strategy

Validating scientific software takes more than checking that the functions run. To hold the Global Fit visualization library to the standards expected of LISA mission infrastructure, the evaluation framework draws on international software engineering standards, domain specific space agency protocols and formal requirements validation.

3.3.1 Standardization and Quality Models

The baseline for software quality is defined by the **ISO/IEC 25010:2023** standard (ISO/IEC 2023). Its role here is not a checklist to pass but a shared vocabulary that keeps the non functional requirements from being chosen arbitrarily. As illustrated in Figure 3.2, the 2023 revision defines nine product quality characteristics and renames the former Usability characteristic to "Interaction Capability", stressing that for a diagnostic interface the quality of the exchange between user and system is a quality attribute in its own right rather than a property subordinate to backend performance. The project does not weight all nine equally. It prioritizes the four that its problem actually stresses. Maintainability and Flexibility govern the architecture, because a mission with a decades long lifetime must absorb new source types and changing upstream formats without rewrites, which is the concern behind NFR-01 and NFR-04. Performance Efficiency governs the data path through two separate guarantees, a rendering peak that stays bounded as the catalog grows (NFR-02) and on-demand reads that let inspection start without loading the whole run (NFR-05), because the library must serve multi gigabyte catalogs on a machine that cannot hold them whole. Interaction Capability governs the visual frontend, because a diagnostic that is misread is a scientific error and not a cosmetic flaw. Each non functional requirement in Table 3.2 therefore names in its rationale the ISO 25010 characteristic it serves, while the obligation to satisfy these qualities at all comes from the ECSS discipline adopted below. The two standards are complementary. ECSS requires that ESA ground segment software exhibit these qualities, while ISO 25010 supplies the taxonomy that fixes which quality each requirement targets.

However, generic software standards must be supplemented by domain specific requirements. For European space missions, software artifacts must comply with the **ECSS Active Standards** (European Cooperation for Space Standardization 2025). These protocols mandate strict traceability and verification processes so that mission critical tools maintain the reliability expected of ESA infrastructure. The evaluation strategy must therefore follow these verification protocols so that the library's architecture remains both maintainable and auditable.

ECSS does not require a master's project to produce the full document suite that a prime contractor would deliver. The standard defines what shall be achieved rather than how the work is organized. It requires each project to tailor its outputs to the software's criticality and complexity (Annex S) (European Cooperation for Space Standardization 2025). Because this library is a ground

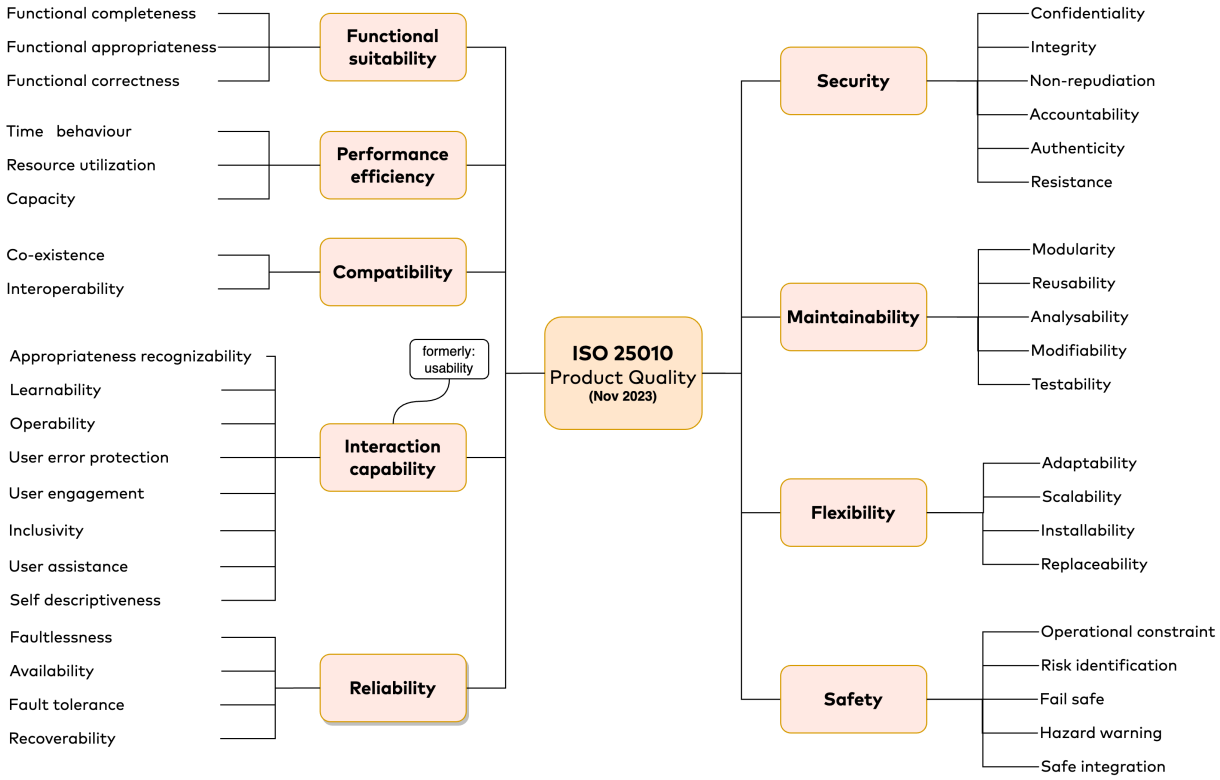

Figure 3.2: **ISO/IEC 25010:2023 Product Quality Model.** The international standard defines nine product quality characteristics, including the newly added 'Safety' and the renamed 'Interaction Capability'. The GlobalFit evaluation strategy uses this framework to assess both the computational performance and the user-system interaction quality. *Adapted from ISO/IEC (2023).*

segment diagnostic tool rather than flight or mission critical software, it falls outside the critical categories that drive the heaviest ECSS verification demands, so the tailoring stays deliberately light. Applying that tailoring here, the essential outputs the standard asks for, a set of software requirements, a design, evidence of verification against those requirements and a record of the result, are carried by this thesis itself rather than by separate administrative documents. The Requirements Engineering of Section 3.2 serves the role of the software requirements specification, the architecture of Chapter 4 serves the role of the software design document and the evaluation of Chapter 6 serves the role of the verification and validation record. This keeps the work aligned with the engineering intent of ECSS-E-ST-40C without manufacturing a paper trail disproportionate to a single library.

A related concern is distribution. The agency dashboards that motivate this work, such as the Earth Observation Dashboard (Anghelea et al. 2023), are built around open science principles so that their outputs and tools can be shared across institutions. The corresponding requirement here is NFR-03: the artifact is an open source Python package installable with pip and free of HPC specific dependencies, so any member of the LISA community can install and run it on a local machine. This is a property of the library rather than of any particular deployment. It holds whether the plots are produced from a script, a notebook or a dashboard.

3.3.2 Requirements Validation and Traceability

To assess the effectiveness of the visualization library during the development lifecycle, the evaluation strategy strictly follows a requirements validation paradigm. In Design Science Research, the quality of an IT artifact is determined by how well it fulfills the specific objectives defined for the solution (Peffers et al. 2007).

Therefore, rather than relying on subjective user satisfaction surveys or generic usability heuristics, the library will be evaluated against a rigid traceability matrix. Every feature of the visualization architecture (e.g., the Data Ingestion Layer, the interactive Sky Maps) must trace back to the core analytical requirements defined by the LISA Global Fit physics working group. This approach ties the artifact's success to whether it meets those traced requirements, concretely whether it can deserialize the HDF5 datasets and abstract visual density without performance failure.

3.4 Design-to-Development Workflow

Moving from a design to working software is not straightforward and needs a structured development lifecycle. Doing it well means balancing strict systems engineering against the exploratory, iterative nature of visualization design.

3.4.1 Lifecycle Models: The Systems Engineering Engine

For mission critical software, the development process is traditionally governed by strict validation protocols. The **NASA Systems Engineering Handbook** (NASA 2016) defines the "Systems Engineering Engine", a recursive model that integrates system design and product realization. As illustrated in Figure 3.3, this engine emphasizes that verification and validation are continuous processes rather than end of lifecycle events. This approach keeps every interface feature traceable to the initial scientific requirements defined by the physics working group, while also supporting alignment with the ECSS standards.

The same handbook defines a second concept that bears directly on this work: Human Systems Integration (HSI). NASA SE treats the human operator not as an external user but as a component of the system, on the grounds that an interface which obscures the system state can cause operational error as surely as a hardware fault (NASA 2016). For this project the operators are the astrophysicists who must read convergence and source structure off the diagnostics, so their capacity to interpret the output correctly is a system property rather than a cosmetic concern. This is the systems engineering basis for the cognitive load, layout and interaction requirements drawn from the visualization literature in Chapter 2: reducing the effort needed to locate and decode a diagnostic is an engineering requirement, because a misread posterior is a scientific error and not merely an inconvenience.

However, even recursive engineering models can be too rigid for the exploratory nature of scientific discovery. Parsons (2022) presents evidence that professional design practice is fundamentally

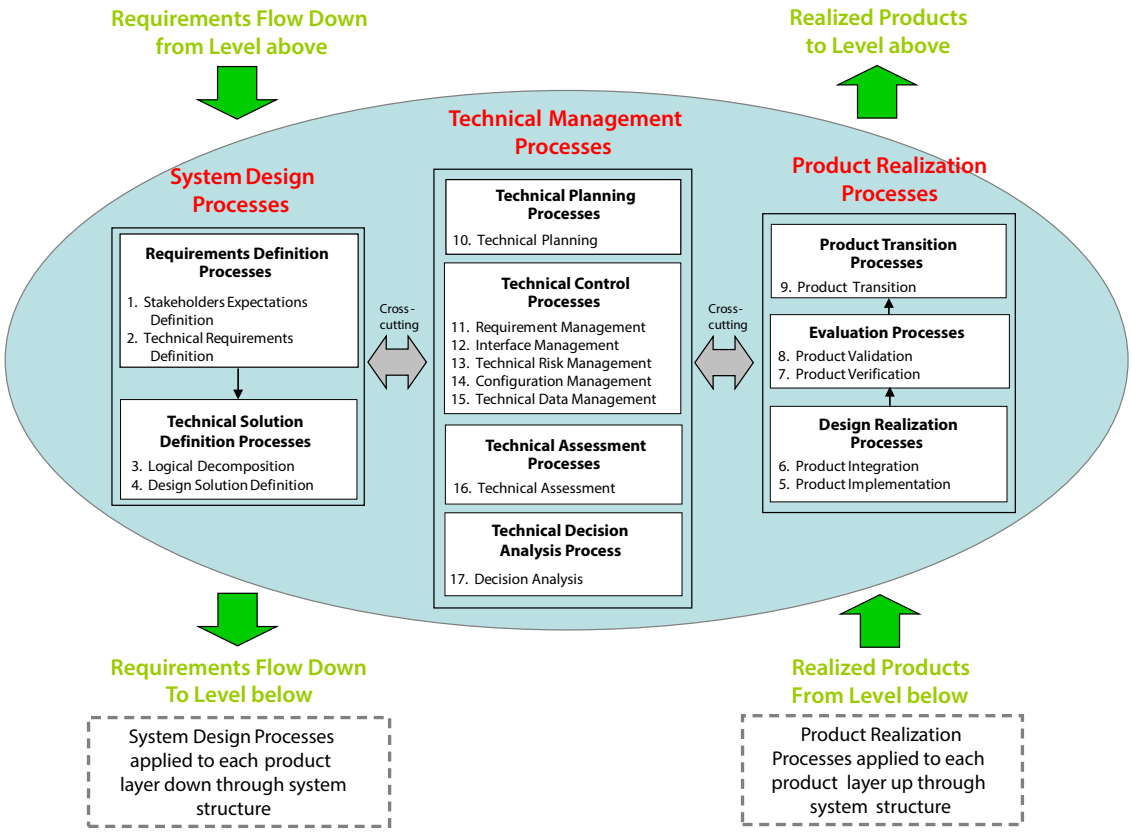

Figure 3.3: **The NASA Systems Engineering Engine.** This diagram illustrates the recursive nature of space system development where System Design (left loop) and Product Realization (right loop) are bridged by Technical Management. This model justifies the use of an iterative yet verified approach. *Source: NASA (NASA 2016).*

"situated": practitioners rely on "reflection in action" and choose methods in the moment as the design evolves in response to the immediate constraints of the data. His study is empirical, which matters for how this project defends its own process. An iterative, precedent driven design loop is not a shortfall in rigor but the observed norm among working visualization designers, so adopting it here is a deliberate choice rather than an absence of method. The practical consequence is a hybrid workflow that combines this iterative prototyping with the formal traceability of the NASA engine.

This hybrid workflow is itself an instance of Hevner et al.'s sixth guideline, Design as a Search Process (Hevner et al. 2004). Hevner frames design as a generate-and-test cycle: when the problem space is too complex to deduce the best artifact in advance, the researcher constructs a candidate, tests it against the requirements and the constraints of the environment and uses what is learned to construct the next candidate. The layout of the diagnostic views could not be settled analytically up front, because the behavior of trans-dimensional MCMC telemetry under real data is only apparent once it is rendered. The visualization modules were therefore developed as a sequence of prototypes, each evaluated against the functional requirements of Section 3.2.2 and against occasional feedback from the astrophysicists operating the pipeline, then revised. This is the search process Hevner describes rather than ad hoc trial and error.

3.4.2 Bridging the Data-Design Gap

A critical challenge in this hybrid workflow is the translation of static design concepts into functional implementation. Walny et al. (2020) identify a systematic friction where traditional design tools fail to capture "data specific" constraints. Their study examines the handoff between a visual designer and an engineer, but the same friction appears within a single designer developer, where it becomes a handoff between the design phase and the implementation phase rather than between two people. A static mockup drawn in a tool like Figma fixes the very things the Global Fit data refuses to hold still. A wireframe of the catalog view assumes a set number of rows, yet the Global Fit returns a different source count from one iteration to the next, so a layout that looks balanced for eight sources overflows for three thousand and empties for a band with none. A sky map sketched with a few dozen tidy points gives no warning of the overdraw that 15.5 million sources produce, which appears only once real data is rendered and which forces the density abstraction of Chapter 2. A posterior drawn as a clean unimodal curve hides the heavy tails, multi modality and label switching that the real chains exhibit and that the axes and binning must accommodate. These are the "messy reality" cases Walny describes. None of them are visible on a static canvas.

To bridge this gap, Walny et al. (2020) recommend a "data centric" specification process. As illustrated in Figure 3.4, this involves explicitly mapping the challenges of data characterization (C1 and C2) directly to the design and development phases. By defining how the visualization should behave under data stress before coding begins, the workflow prevents the common disconnect between design intent and technical realization.

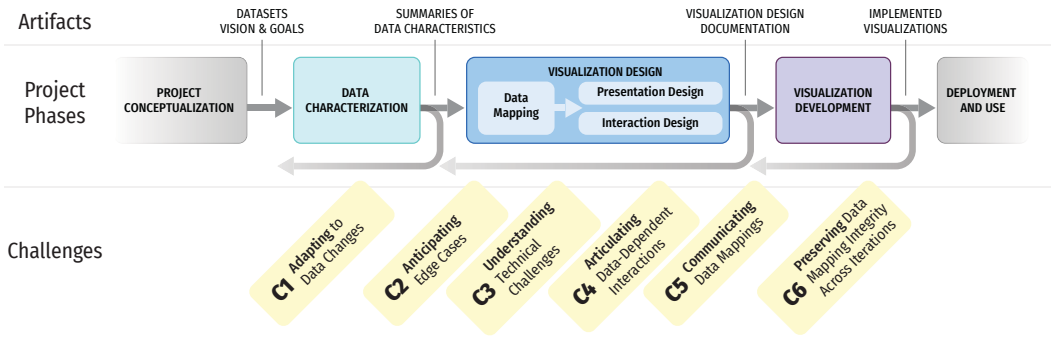

Figure 3.4: **The Data-Driven Visualization Design Process.** Unlike standard UI workflows, scientific visualization requires a process where data characterization directly informs visualization design. The challenges highlighted (C1-C6) demonstrate where the translation from design to code typically breaks down due to data variability and edge cases. *Source: Walny et al. (2020).*

3.4.3 Empirical Data Discovery and Schema Inference

Before the formal software architecture could be designed, the first practical challenge was executing the data characterization step (C1 and C2 in Figure 3.4). The upstream pipeline samplers output their results into a complex, nested directory structure (the `gfrun` directory) containing dozens of HDF5 files, numerical arrays and configuration logs. However, at the start of this project, there was

no formal API documentation, strict schema definition or data dictionary available that a software engineer could use to build a reliable data ingestion layer.

To bridge this gap in the requirements engineering phase, an automated data discovery process was developed. Crawler scripts were deployed to recursively traverse the `gfrun` directory and introspect the metadata of every HDF5 file using the `h5py` library. These scripts mapped out the internal group hierarchies, dataset shapes, data types and the presence of attached attributes. For the trans-dimensional MCMC chains, the scripts specifically characterized how the array dimensions fluctuated to accommodate the variable number of Galactic Binaries per iteration.

The output of this automated inference process was compiled into a formal engineering artifact: the `lisa_data_dictionary.txt`. This file became the strict source of truth for the early software engineering process, explicitly mapping the physical location of the data (e.g., `<band>/new_source/chain_0.npy`) to its astrophysical meaning.

This characterization directly informed the Domain-Driven Design (DDD) approach and the creation of the Repository Pattern described in Chapter 4. The Data Dictionary provided the exact blueprints needed to map scattered arrays into cohesive Python domain objects. That this empirical inference was needed at all is what justified the abstract Data Ingestion Layer (NFR-04). Because the legacy schema was opaque and actively evolving into the formal L2-to-L3 Data Model, isolating the parsing logic behind an interface kept the core visualization domain protected from breaking upstream data format changes.

3.5 Experimental Process and Evaluation Design

The validation of the visualization library will focus on verifying that the artifact meets the formal requirements derived from the LISA mission's data constraints and from the research objectives defined in Section 2.5.

3.5.1 Evaluation Framework

The evaluation aligns with Activity 5 of the DSRM process model, comparing the objectives of the solution to the actual observed results (Peffers et al. 2007). The framework is based on:

- **Requirements Traceability:** showing that every architectural layer fulfills a specific computational objective, the traceability ECSS-E-ST-40C requires of ESA mission software.
- **Visual Heuristics Verification:** Inspecting the generated interactive plots to confirm they implement the visualization principles established in Chapter 2 (e.g., mitigating overdraw via Splatterplots-inspired density abstraction).
- **Software Quality:** evaluating the library against the two ISO/IEC 25010 characteristics this problem stresses, *Performance Efficiency* for parsing multi-gigabyte datasets and *Interaction Capability* for the visual frontend.

3.5.2 Verification Approach

Verification is carried out on the **LDC2a "Sangria" dataset** along two lines:

1. **Bounded Memory Verification:** Confirming that the Data Ingestion Layer parses multi gigabyte HDF5 catalogs by streaming them in bounded blocks, so that the library loads and renders the full resolved population without exhausting memory (NFR-02). The claim under test is the architectural one, that peak memory does not grow with catalog size by construction.

2. **Rendering and Visual Verification:** Confirming that the final visual outputs apply the required density reduction heuristics when plotting millions of trans-dimensional points, so that the rendered display stays legible under overdraw.

A quantitative benchmark of ingestion time and rendering framerate against catalog size would put numbers on this behaviour. That measurement is left as an evaluation to run on the maturing library, as discussed in Section 6.3, because the property this work defends is the one the architecture guarantees rather than a particular timing on a particular machine.

3.6 Expected Results and Relevance

If successful, this work will produce the contributions set out in Section 1.6: a reusable reference architecture, a validating Python library and a literature-grounded dashboard layout that together replace the current ad hoc plotting workflow with an engineered foundation for the mission's visualization infrastructure.

Chapter 4

System Architecture

This chapter presents the reference architecture that Section 1.6 names as the central contribution: a technology agnostic design for domain specific scientific visualization systems operating over trans-dimensional MCMC data. Every architectural decision documented here is traceable to the requirements defined in Chapter 3 and is grounded in the software engineering literature cited at the point each decision is made. Each module boundary and dependency direction can be defended on theoretical grounds rather than personal preference.

Figure 4.1 presents a high level conceptual overview of the system context. The visualization library sits downstream of the central catalog produced by the inference engine. It reads the HDF5 outputs, deserializes them into domain models and feeds them to an interactive diagnostic suite.

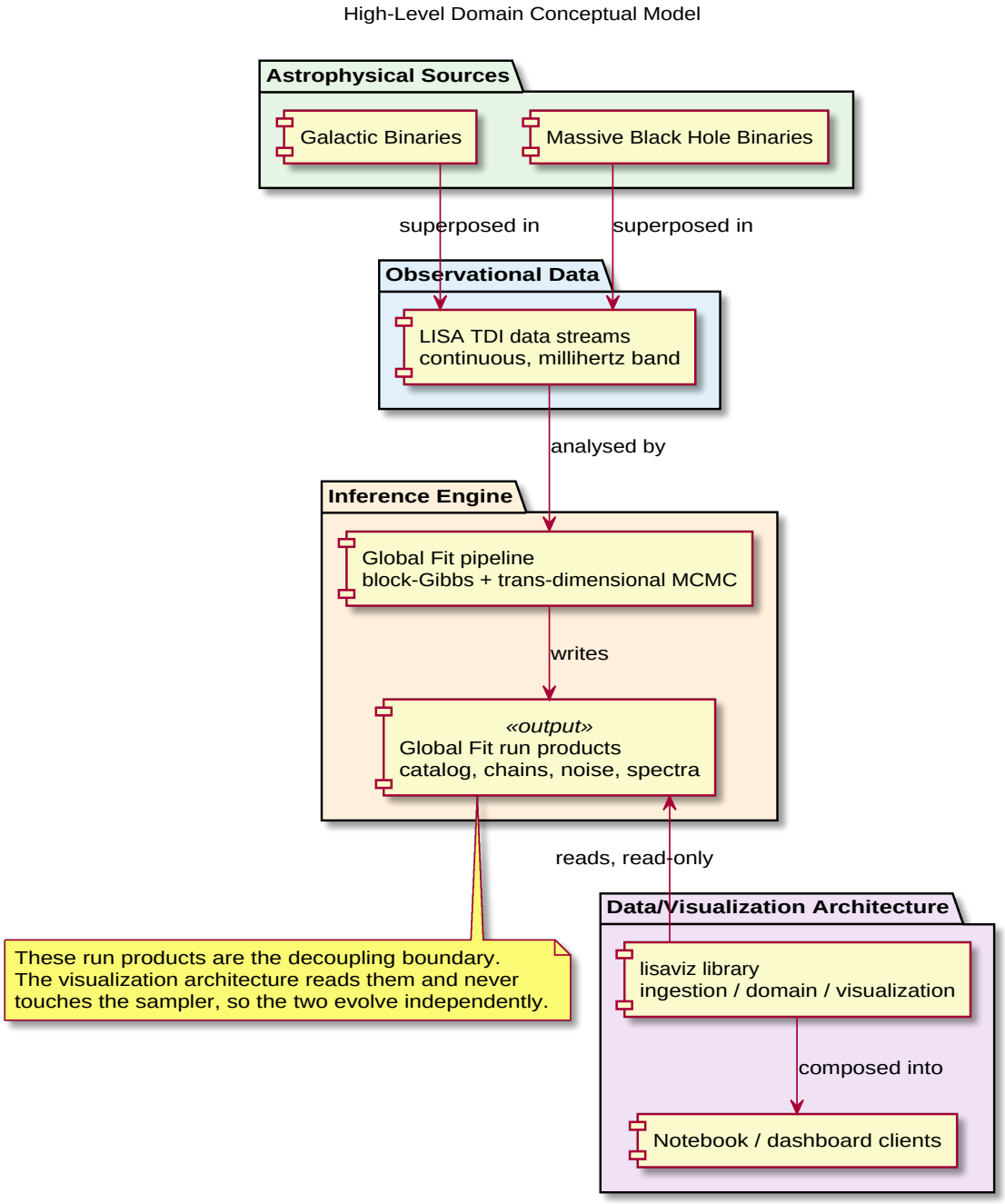

Figure 4.1: **High-Level Domain Conceptual Model.** The system is organized into four packages: Observational Data, Astrophysical Sources, Inference Engine and Data/Visualization Architecture. The visualization library is decoupled from the inference engine through the central catalog.

4.1 Quality Attribute Requirements

Before defining the structure of the system, it is necessary to establish what the architecture must achieve. The ISO/IEC 25010 quality model adopted in Chapter 3 fixes which qualities matter. Its companion measurement standard ISO/IEC 25023 (ISO/IEC 2016) supplies the way to

make them measurable, so the design is constrained by quality characteristics decomposed into subcharacteristics and then into measures read off the artifact. This turns subjective goals ("the system should be fast") into objective, falsifiable requirements.

Table 4.1 lists the quality characteristics that govern the architecture.

Quality Characteristic	**Subcharacteristic**	**Measure & Target Value**
Performance Efficiency	Resource Utilization	Peak resident memory during streaming rendering shall stay bounded by the block size, not the source count. Ingestion parses the 0.6–0.7 GB posterior batches sequentially (LISA DDPC 2026b).
Interaction Capability	Operability	Design target: Sky Map interaction latency (e.g., panning, zooming) below 1.0 s at mission scale (∼15.5M sources (LISA DDPC 2026a)) via Splatterplots-inspired density reduction.
Maintainability	Modifiability	An upstream format change (L2-to-L3 schema) shall require 0 modifications to the Core Domain and exactly 1 new Adapter.
Maintainability	Reusability	The Core Domain shall contain 0 imports from infrastructure libraries (`h5py`, `plotly`, `arviz`).
Maintainability	Testability	All domain and visualization logic shall be testable without real HDF5 files.

Table 4.1: Quality Attribute Requirements mapped to ISO/IEC 25010 characteristics with ISO/IEC 25023 measures.

The right hand column of Table 4.1 lists these as ISO/IEC 25023 measures read directly off the artifact rather than judged by eye (ISO/IEC 2016). The cross-layer import count in the Core Domain is an internal measure, a static property the build verifies by inspecting the source, while the peak resident memory during ingestion is an external measure, observed by running the code. The 1.0 s interaction latency target derives from Miller's response time limits (Miller 1968) for keeping a user's flow of thought intact during interactive tasks, thresholds that Nielsen later popularized. Holding the internal measures apart from the external behaviour they predict is what lets the evaluation in Section 6.2 replace subjective claims about code quality with measured values.

Every architectural decision described in the following sections exists to satisfy one or more of these measures. When a decision is introduced, the corresponding quality characteristic is stated.

4.2 Layered Component Model

The library is demonstrated on the LDC2a Sangria dataset, which provides known injected sources to check that its visual outputs are correct. The scale requirements come instead from the Mojito

dataset, far larger in data volume, that the pipeline must eventually handle. Sangria supplies the ground truth and Mojito the scale, so the architecture is built to meet both.

The architecture separates the data processing backend from the interactive frontend, the same split that national space agencies adopt in systems like the Space+ Data Dashboard described by Jr., Cruz, and Blanco (2024). Keeping the two apart means the heavy computational demands of the LISA pipeline do not slow down the user interface. Therefore, the system is organized as a layered architecture composed of three principal layers: the Data Ingestion Layer, the Core Domain Layer and the Visualization Layer. These three layers correspond directly to the canonical enterprise triad that Fowler (2002) identifies in any sizable application: Presentation, Domain and Data Source. The Visualization Layer is the Presentation layer, the Core Domain Layer is the Domain and the Data Ingestion Layer is the Data Source. Fowler's argument for keeping these layers distinct is that without firm boundaries logic leaks across them: presentation code starts performing domain calculations, or domain code starts issuing storage queries, until the system becomes difficult to test and to change. The boundaries enforced here block both failure modes. The astrophysical computations cannot leak into the rendering code because the Visualization Layer holds no domain math. The HDF5 access logic cannot leak into the domain because the Domain imports nothing from the Ingestion Layer.

Applying Parnas's Information Hiding criterion (Parnas 1972), the most volatile design decision in the Global Fit context is the structure of the HDF5 files produced by the upstream Global Fit pipeline. The parameter naming conventions, the group hierarchy and the storage chunking strategy are all subject to change as the pipeline evolves. By encapsulating all knowledge of the HDF5 format inside the Data Ingestion Layer, the architecture confines a change to the file format to exactly one module, directly satisfying the Modifiability measure from Table 4.1.

Applying the Dependency Rule (Martin 2017), source code dependencies point strictly inward. The Core Domain Layer sits at the center and imports nothing from the outer layers. The Data Ingestion Layer depends on the Domain (it must know how to construct domain objects) and the Visualization Layer depends on the Domain (it must know how to read domain objects for rendering), but the Domain itself is entirely self contained. This satisfies the Reusability measure: because the Domain has no knowledge of `h5py` or `plotly`, it can be reused in any context without pulling in infrastructure dependencies.

The rendering stack is the most volatile part of the system. The library currently renders through Matplotlib and Plotly. The monitoring dashboard envisaged in Chapter 7 may eventually sit on a different frontend altogether. Because dependencies point only inward, any such change is confined to the Visualization Layer: a new plotting backend, or a future web frontend, can be adopted or discarded without touching the Galactic Binary domain logic, since that logic neither imports nor references the rendering code. The lighter alternative is a branch on the backend name in each plotting function, an `if plotly ...else matplotlib` repeated wherever a figure is drawn, so adding a third backend means editing every one of them. Routing the backends through a Strategy interface confines that choice to the concrete strategies, so the rendering stack can change without the plotting functions changing with it.

This layered, object oriented approach breaks away from fragile, ad hoc scripting and has precedent in the Python scientific ecosystem. Mayavi (Ramachandran and Varoquaux 2011), for example, keeps its core domain logic separate from the graphical interface so the two can change independently. The layered separation enforced here (NFR-01) serves the same end: the rendering logic stays independent of the underlying data pipeline, because the Domain imports neither the storage code nor the rendering code.

These two mechanisms share a common root in Dijkstra's separation of concerns (Dijkstra 1982). Keeping the Core Domain agnostic to both the HDF5 storage format and the rendering backend means the astrophysical logic, the definition of a Galactic Binary and the relationships between its parameters, can be reasoned about and tested for its own internal consistency without being entangled with disk layout or plotting code. Information hiding decides where the module boundaries fall, the dependency rule fixes the direction in which they may be crossed and separation of concerns explains why that isolation is necessary in the first place.

Figure 4.2 presents the component diagram of the architecture, showing the three layers and the direction of their dependencies.

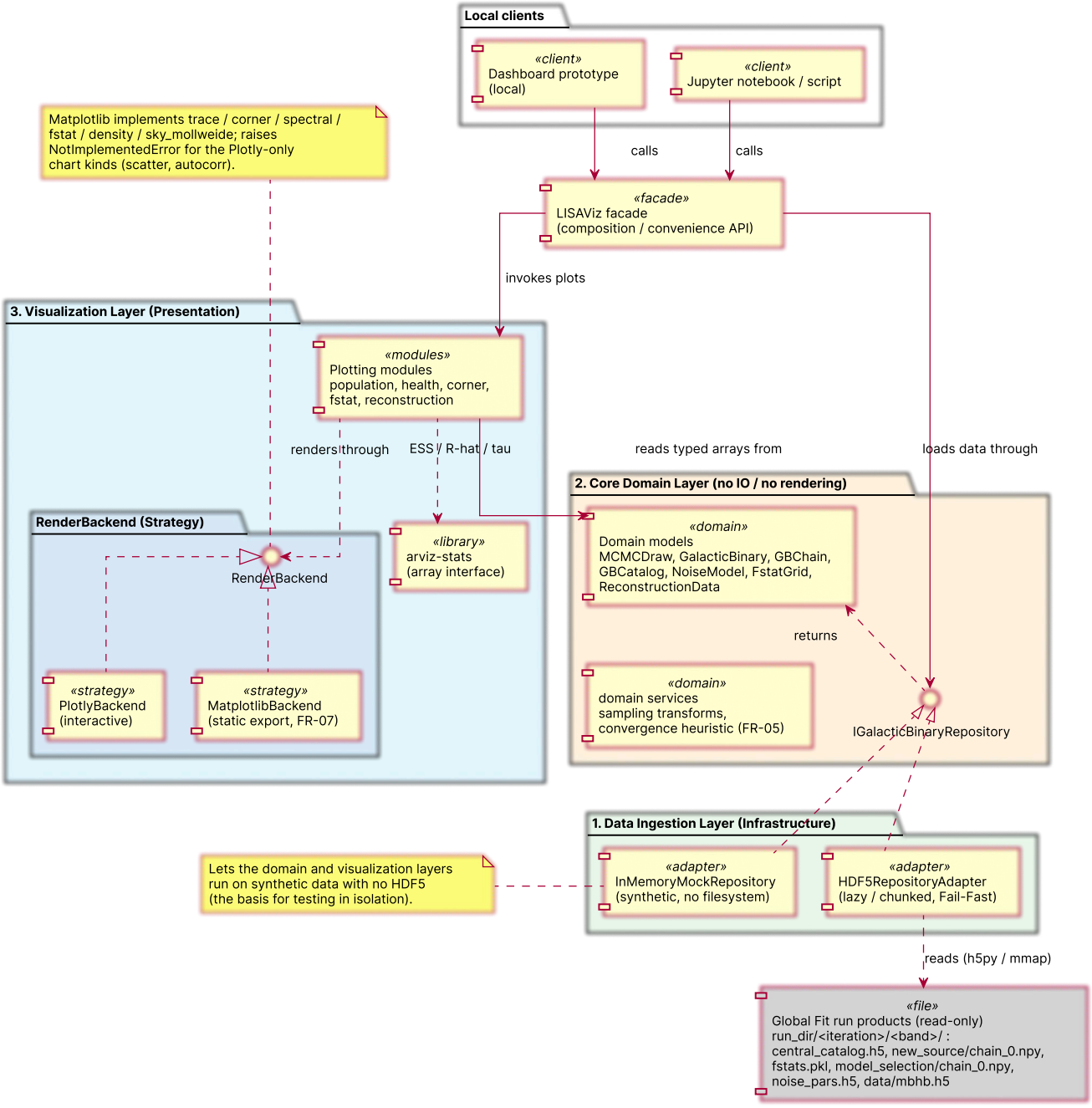

Figure 4.2: **Layered component architecture.** Dependencies point inward to the Core Domain, which imports neither the ingestion nor the rendering code. The Data Ingestion Layer provides two interchangeable adapters, the `HDF5RepositoryAdapter` and an `InMemoryMockRepository`, both implementing `IGalacticBinaryRepository`. The Visualization Layer renders through the `RenderBackend` Strategy, Plotly for interaction and Matplotlib for static export. The `LISAViz` facade composes these for local clients such as notebooks.

Using Stevens's coupling taxonomy (Stevens, Myers, and Constantine 1974), the coupling between layers is classified as Data Coupling, the weakest and most desirable form in its seven level hierarchy. Layers communicate only through typed domain objects (Value Objects and Entities),

never through shared global state or raw data structures. The Visualization Layer receives sanitized arrays extracted from these domain objects and never references the internal state of the HDF5 file handlers in the Ingestion Layer, so a change to the file handling logic cannot propagate into the rendering code. The same taxonomy applies within the Visualization Layer rather than only across its boundary. Each plotting component exhibits Functional Cohesion: it executes a single rendering task, such as producing a trace plot or a corner plot or a sky map, without relying on shared mutable state. Pairing data coupling across layers with functional cohesion within modules is the structural property that satisfies the Modifiability measure from Table 4.1, because it confines any change to the module that owns the affected decision.

4.3 Domain Model

The Core Domain Layer applies the Domain Driven Design (DDD) methodology. Rather than passing raw arrays of floating point numbers through the system, the architecture defines a set of typed domain objects that encode the physical meaning of the data. These objects carry what Evans calls a Ubiquitous Language (Evans 2003): a shared set of terms grounded in the domain rather than in the storage format. The model names quantities the way the physicists do, as `frequency`, `amplitude` or `declination`, instead of opaque positional labels such as `column_3` or `array[:, 4]`. This vocabulary was not invented for the software. It was taken from the parameter definitions used by the LISA working group and the data dictionary compiled during the schema inference phase (Section 3.4.3), then refined through occasional consultation with the astrophysicists operating the pipeline. Aligning the domain objects with the terms the scientists already use keeps the visualization controls and the public API readable to a physicist and removes a translation step between the physics and the code.

Each of Evans's three domain object types appears in this model.

Entities are objects defined by their identity rather than their attribute values. In this system, the `GalacticBinary` is an Entity: two binary sources with identical parameter values at different positions in the MCMC chain are not the same object, because their identity is tied to their role in a specific draw.

Value Objects are immutable objects defined entirely by their attribute values. The `IntrinsicParameters` and `ExtrinsicParameters` are Value Objects. Two `IntrinsicParameters` instances with identical values are semantically interchangeable. Modeling them as typed objects rather than naked arrays provides type safety at the API boundary: it becomes impossible to accidentally pass extrinsic parameters where intrinsic parameters are expected, satisfying the "hard to misuse" principle articulated by Bloch (2006).

Aggregate Roots control access to clusters of related objects. The `MCMCDraw` is the Aggregate Root for the source catalog within a single iteration. Because the Global Fit infers a varying number of sources rather than fixing it, the number of resolved sources changes from draw to draw. The model represents that count directly, an Aggregate Root holding a variable-length collection of `GalacticBinary` entities whose parameters are still reached by name through the

`IntrinsicParameters` and `ExtrinsicParameters` Value Objects. Modeling the source count as a first class domain fact keeps named, typed access without committing the domain to a fixed rank, so a draw with three sources and a draw with three thousand are the same kind of object differing only in length. Keeping each draw at its true source count aligns the model with the domain, where that count is itself the quantity that varies. A generic statistical container such as the one ArviZ operates on (Section 2.4.2) reasons over named axes rather than over Galactic Binaries, so the domain meaning of the count lives in the model here rather than in the storage layout.

Figure 4.3 presents the detailed UML class diagram of the domain model.

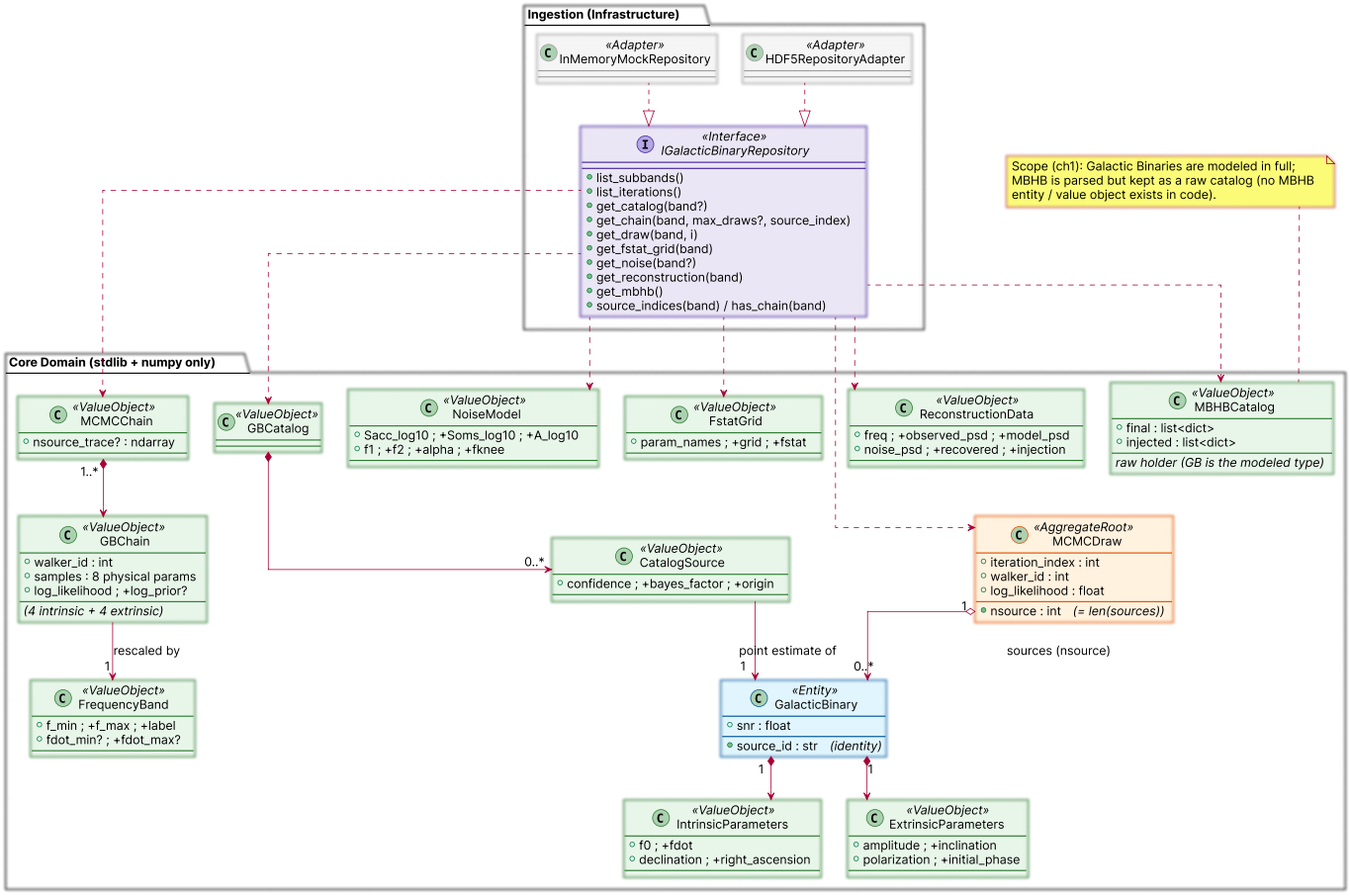

Figure 4.3: **Core domain model.** The `MCMCDraw` Aggregate holds a variable-length collection of `GalacticBinary` Entities, each composed of `IntrinsicParameters` and `ExtrinsicParameters` Value Objects, so the source count is represented directly as a variable-length collection. The parameter estimation chains (`GBChain`, `MCMCChain`) carry the 8 sampled parameters that the resolved `GBCatalog` also records as point estimates. The `IGalacticBinaryRepository` interface and its two adapters sit at the layer boundary. Galactic Binaries are modeled in full while `MBHBCatalog` stays a raw holder, matching the scope of this work.

4.4 The Repository Interface and Dependency Inversion

A central architectural challenge is how to get data from the HDF5 files (which live in the Infrastructure layer) into the Domain (which must not import `h5py`). The simpler alternative is a plain loader function that returns a `pandas` DataFrame, which is what the current scripts do. That couples every caller to `pandas` and to the exact HDF5 column names the loader exposes, so the volatile decision Parnas isolates leaks straight into the Domain. That coupling is what the Reusability measure counts as cross-layer imports, which the loader would drive above zero the moment the Domain read its DataFrame. The Repository pattern (Fowler 2002) keeps the count at zero.

The Domain Layer defines an abstract interface called `IGalacticBinaryRepository`. This interface specifies a contract: "give me draw number N and I will return an `MCMCDraw` object." The interface lives inside the Domain, not inside the Ingestion Layer. The Data Ingestion Layer then provides a concrete implementation (`HDF5RepositoryAdapter`) that satisfies the contract by reading the HDF5 file. This applies the Dependency Inversion Principle: data flows from the outer layer inward, while the source code dependency also points inward, because the outer layer imports and implements the Domain's interface.

This design has three practical consequences. First, it satisfies the Testability measure: because the Domain and the Visualization Layer depend only on the abstract interface, they can be tested using a simple `InMemoryMockRepository` that returns synthetic domain objects without ever touching the filesystem. Second, it satisfies the Modifiability measure: the official L2-to-L3 Data Model fundamentally changes the expected output format of the pipeline (e.g., batching posteriors into files of 500 sources, requiring a master index file and migrating to `snake_case` parameter naming) (LISA DDPC 2026b). If the LISA consortium migrates storage formats, the only code that changes to support these breaking updates is the concrete adapter in the Ingestion Layer. The Domain and Visualization layers remain completely untouched. Third, it guarantees architectural expandability (NFR-04): while the current implementation focuses exclusively on Galactic Binaries, adding a future source type (MBHB, EMRI or Noise) means writing one new adapter class, with no change to the core visualization domain.

Fowler also warns of the "`all()` danger": because a Repository presents itself as an in memory collection, a careless caller might request the entire dataset at once, crashing the system's memory. This risk is extreme in the LISA context: uncompressed L2-to-L3 pipeline outputs reach 20 GB per submission (LISA DDPC 2026b), while the underlying Mojito Light dataset exceeds 1.3 TB (LISA DDPC 2026a). To keep this in check, the `IGalacticBinaryRepository` offers bounded reads for the large data, returning chains within a specified draw range together with the catalog in fixed-size blocks for the macroscopic views, while a full read of the catalog stays available for the small catalogs whose source count sits well below the memory budget.

The concrete implementation limits memory consumption through two distinct patterns. First, to handle macroscopic catalog views, it uses the Iterator pattern to stream the data sequentially in fixed-size blocks. This pattern flows through the three architectural layers: the ingestion layer slices the storage into chunks, the adapter translates them into domain models and the visualization

layer consumes them sequentially. This separates concerns, as the visualization loop knows nothing about the underlying storage mechanics. Because each chunk is discarded after use, peak memory is bounded by the block size.

Second, for the underlying disk access, it relies on the Virtual Proxy pattern provided by the `h5py` interface and NumPy memory-mapping. As Collette (2013) explains, HDF5's subsetting and partial I/O capabilities are designed precisely for this kind of out-of-core evaluation. The adapter treats each HDF5 dataset on disk as if it were a complete in-memory NumPy array, but the Virtual Proxy lazily pages the actual float values from disk on-demand only when a specific slice is requested, for example the catalog rows for a single frequency subband, without materializing the rest of the file. The Galactic Binary chains are reached the same way through memory-mapping of their `.npy` files, so only the requested draw range is read from disk. When a physicist inspects one source or one subband, the Ingestion Layer fetches exactly those rows from the Global Fit output products and nothing more. This architectural choice aligns directly with the official DDPC software engineering strategy, which explicitly implements "chunked" processing to limit RAM consumption when reading these multi-gigabyte files. On the bounded reads this satisfies the Performance Efficiency measure from Table 4.1, since peak memory is then set by the block size rather than the total file size. Section 4.5 sets out where this streaming generalizes and where it does not.

The adapter only ever reads from the HDF5 files and never writes back to them. This read-only constraint follows the data management practice recommended by Wilson et al. (2016), which holds that raw datasets must be treated as immutable to preserve reproducibility. Treating the telemetry as immutable also bounds the failure modes of the Ingestion Layer: a parsing defect can produce an incorrect plot, but it cannot corrupt the multi gigabyte posterior chains that the pipeline spent compute hours generating. This constraint already applies in the current setup, where the library runs separately from the pipeline and a physicist downloads the summary catalogs to a local machine. It becomes a hard safety requirement if the library is later embedded inside the Global Fit environment and shares a filesystem with the live run outputs, because at that point the same code path has the opportunity to overwrite the data it is meant to visualize.

4.5 Bounded Memory

The bounded reading just described lowers peak memory only where the view it feeds is a reduction of the data rather than the data itself. A view that can be built from a fixed-size summary, one that absorbs the catalog block by block and never grows, gains the full benefit, since the blocks are read, folded in and discarded one at a time. A view that needs the whole data vector in hand gains nothing from streaming, because the vector has to be resident before the computation can run at all. The architecture therefore applies streaming exactly where it generalizes and leaves it out where it would add complexity for no benefit, which is the deliberate weighing of cost against gain that Hevner's design as a search process calls for (Hevner et al. 2004).

The macroscopic catalog views are where it generalizes. The sky map and the waterfall are the same reduction in two coordinate frames, a fixed grid of density together with a bounded outlier reservoir, a fixed-capacity store that keeps the most uncertain sources as discrete points. The bound on that store is what keeps the memory flat as the catalog grows, so the scaling claim holds only if the store keeps the globally most uncertain sources rather than whichever ones the stream reaches first, which is the condition the streaming sky map meets by construction (Section 5.5). This takes the Splatterplots principle of preserving the outliers while abstracting the dense regions and reads it as an in-memory operation rather than only a screen one (Mayorga and Gleicher 2013). The dense region mechanism is deliberately adapted, because Splatterplots blends kernel density contours over a multi-class scatter whereas the sky map carries a single continuously coloured population, so the dense regions are reduced by hexagonal binning instead. The reduction is therefore Splatterplots-inspired rather than the original technique, a point Section 6.3 returns to when it measures the rendered result. Because both views share this shape, the same block-wise reduction serves both, so abstracting density in memory before rendering is the general strategy for every catalog scale view rather than a trick for one plot. The chain views are where it does not generalize. A trace, a corner, an autocorrelation or the convergence diagnostics are computed from the whole per-parameter chain, since the effective sample size, the R-hat statistic, the autocorrelation time and the kernel density estimates each read the entire vector, so there is no fixed-size summary to stream into. Those reads are already bounded by the draw range and the memory-mapping of the chain files, so streaming would buy nothing. The F-statistic grid and the noise and MBHB reads are left unstreamed for the same reason, the first decoded whole as a small pickled grid and the second only a single row or a small table.

Figure 4.4 shows the two paths side by side. The object path holds one set of objects per source so its cost grows with the catalog, while the streaming path folds each block into a fixed grid and a bounded reservoir then discards it, so its peak is set by the block rather than the source count.

The two heavy population views, which are the only paths whose cost would otherwise grow with the source count, are therefore the two that stream, while the paths left unstreamed are either already bounded or small enough that the question does not arise. Chapter 6 measures the population views holding flat as the catalog grows.

4.6 API Design Principles

The public interface of the library is governed by the API design principles of Bloch (2006) and Blanchette (2008).

The “Fail Fast” principle requires errors to surface as early as possible. The upstream pipeline writes HDF5 with automatic chunking (`chunks=True`), whose layout can be misaligned for sequential draw reads and cause “chunk thrashing”, so the `HDF5RepositoryAdapter` inspects the chunk metadata at construction and raises immediately on an incompatible layout rather than failing silently at render time (Collette 2013).

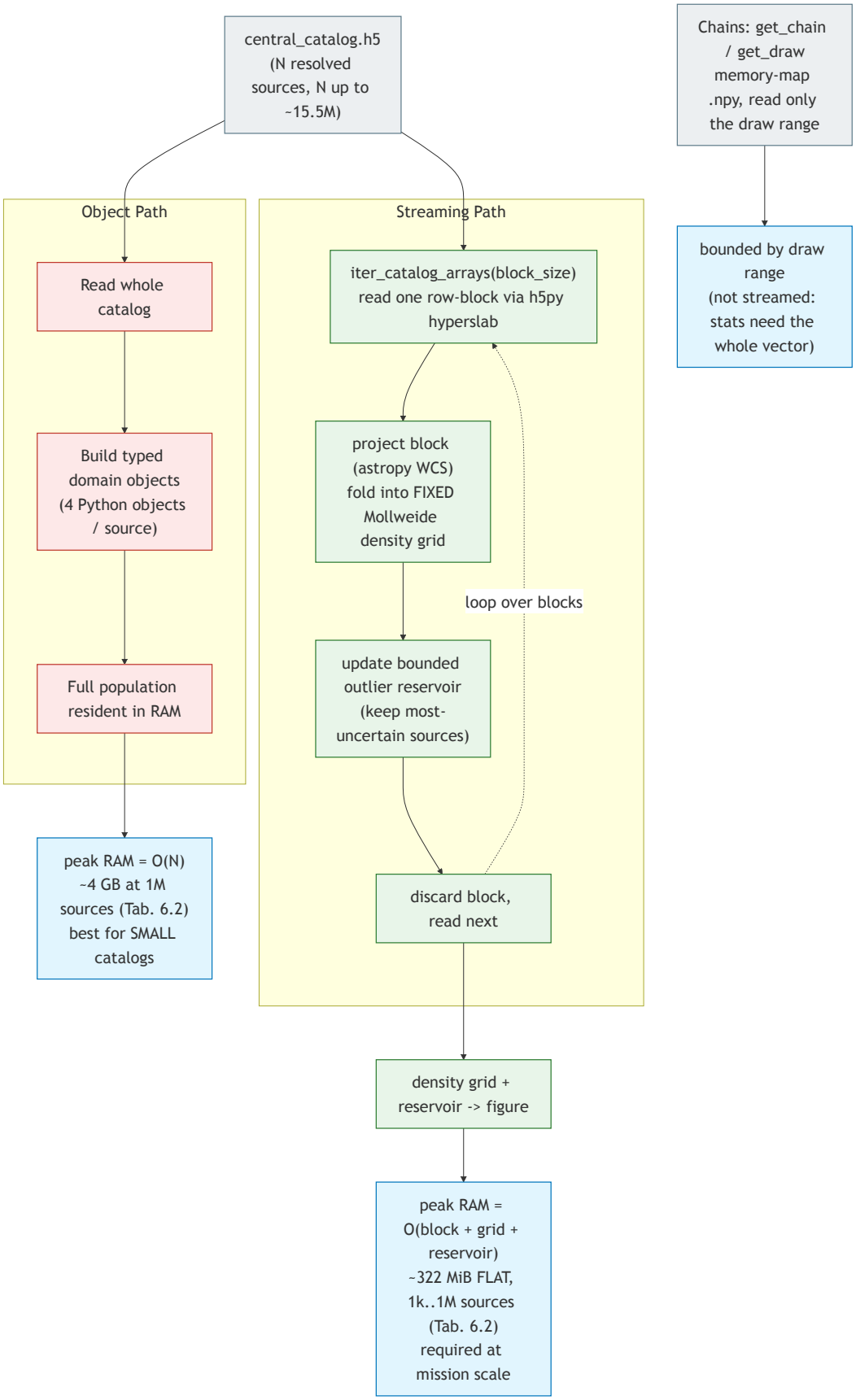

Figure 4.4: **Bounded memory in the population views.** The object path materializes one set of objects per source so its peak grows with the catalog. The streaming path reads the catalog in fixed blocks, folds each into a fixed density grid and a bounded outlier reservoir then discards it, so its peak is set by the block size and stays flat as the catalog grows. The chain reads are bounded separately by the draw range through memory-mapping.

The "hard to misuse" principle is satisfied by the Domain Model itself. Because the plotting functions accept typed Value Objects (`IntrinsicParameters`, `ExtrinsicParameters`) rather than generic numerical arrays, it is not possible for a user to accidentally swap the order of parameters or pass data in the wrong coordinate system. The type system enforces correctness at the interface boundary. This also follows Blanchette's recommendation to favor property based access over long positional argument lists (Blanchette 2008). A `GalacticBinary` is not constructed from eight bare floats passed in a fixed order, where transposing right ascension and declination would silently produce a valid but physically wrong source. It is composed from named Value Objects whose fields are addressed as explicit properties (`frequency`, `declination` and so on), so each quantity is identified by name rather than by position. Grouping the parameters into `IntrinsicParameters` and `ExtrinsicParameters` keeps the construction site readable despite the dimensionality of the data and removes the boilerplate of threading many loose arguments through the call chain. This philosophy directly mirrors the successful API strategy used by Mayavi (Ramachandran and Varoquaux 2011). Mayavi's `mlab` interface intentionally abstracts a highly complex Visualization Tool-Kit (VTK) pipeline behind simple, one-line scripting commands, answering common scientific use cases consistently without requiring users to understand the underlying mechanics. Ramachandran and Varoquaux (2011) describe their goal as building reusable, general purpose abstractions that hide a heavy backend behind a Pythonic scripting interface. The same goal drives this library. The HDF5 ingestion, the trans-dimensional chain handling and the rendering backends are all reachable through high level calls that a physicist can run directly in a Jupyter notebook, without touching `h5py` handles or chunk layouts. The dashboard layout proposed in Chapter 5 is one way to compose this interface and not the only way to use it. The same plotting functions that the layout arranges can be called on their own during interactive analysis, which keeps the library rather than any one interface as the primary artifact of this thesis.

Bloch also argues that implementation details must not leak into the public interface, because anything the interface exposes becomes a permanent commitment that callers will come to depend on (Bloch 2006). This is not a separate justification for the layer boundary. It is the same boundary that information hiding (Section 4.2) and the Repository contract (Section 4.4) already establish structurally, stated here from the caller's side: a physicist using the library sees only domain objects and plotting calls, never `h5py` handles, chunk shapes or disk offsets. Expressing the principle at the API level adds one concrete consequence to that structural argument. Because the public interface commits to nothing about how the data is stored, the anticipated migration to the L2-to-L3 data model, with its batched files and `snake_case` renaming, can be absorbed entirely inside the adapter without changing a single call that downstream code has already written.

Finally, the "when in doubt, leave it out" principle justifies the deliberately minimal scope of the initial release. The library's public API is organized around four visualization modules (MCMC Health, Posterior Inference, Signal Reconstruction and Population Diagnostics), each exposing a small number of focused plotting functions. This restraint is a design decision, not a limitation: a minimal public API can always be expanded in future versions, but functionality that has been

published cannot be retracted without breaking downstream users.

4.7 Design Patterns Applied

Several established architectural patterns are applied in this architecture, using the object oriented design patterns cataloged by Gamma et al. (1994) and Domain-Driven Design principles to handle fine grained structural challenges. Table 4.2 summarizes each pattern, its location in the system and the architectural claim it supports.

Not every pattern here carries the same weight. The Repository, the Adapter, the variable-length Aggregate and the Strategy each isolate a volatile decision whose simpler alternative was shown to fail above, on a measured cost in the first three. The rest are conveniences and are reported as such. The **Facade** lets a physicist call `sky_map()` rather than coordinating file reading, physics math and rendering by hand, which shortens the call site but isolates no volatility the layer boundaries do not already isolate. The **Factory Method** hides the assembly of a mock or real environment behind one call. **Dependency Injection** passes the repository into the facade's constructor, which is load-bearing only as the mechanism the Repository needs to swap the real adapter for the `InMemoryMockRepository` under test, so it serves Testability rather than standing as its own claim. The **Repository** does double as a Domain Factory at the ingestion boundary, translating raw data arrays into constructed `GalacticBinary` models so the client never extracts HDF5 columns by hand, but that is the Repository's coupling argument again rather than a separate one.

At the domain level, the architecture relies heavily on Domain-Driven Design constructs. The **Value Object** pattern is applied to structures like `IntrinsicParameters`, which are implemented as frozen Python dataclasses. Because they are immutable and lack independent identity, two instances with identical parameters are considered equal, eliminating side-effect bugs. Conversely, the **Entity** pattern is applied to objects like the `GalacticBinary` model, which maintains a distinct physical identity (its `source_id`) even as its parameters evolve across pipeline iterations.

The **Adapter** pattern is used within the Data Ingestion Layer, where it acts as a protective boundary rather than a mere format converter. It translates the pipeline's heterogeneous storage formats (HDF5, NumPy arrays and Pickle files) into the stable **Repository** interface defined by the Domain. The concrete `HDF5RepositoryAdapter` presents the Domain's `IGalacticBinaryRepository` interface on the inside while speaking the respective format APIs (`h5py`, `numpy`, `pickle`) on the outside, keeping the Domain entirely isolated from storage mechanics.

The **Strategy** pattern is applied in the Visualization Layer, where each rendering backend is an interchangeable algorithm behind a common interface. The plotting modules act as the context: they translate domain models into neutral coordinate arrays and delegate the drawing to the chosen strategy. The current implementation provides two such strategies: Matplotlib for static export and Plotly for interactive inspection. Because the plotting functions pass plain arrays rather than backend-specific objects, the rendering strategy can be substituted without the Core Domain or the

calling code being aware. New visualization technologies can then be integrated as new concrete strategies if they become the industry norm.

The **Iterator** and the **Virtual Proxy** are load-bearing again, since both bound memory: the Iterator yields sequential chunks of the catalog and the Virtual Proxy memory-maps the file to page float values from disk only when sliced, the mechanism behind the Performance Efficiency measure. **Lazy Initialization** (via PEP 562) is housekeeping. It defers importing the heavy rendering libraries until a plot is requested, which speeds the test suite but could be dropped without touching the architecture.

Pattern	**Location**	**Architectural claim**
Facade (GoF)	`LISAViz` root class	Provides a unified, simple interface to the complex subsystems.
Adapter (GoF)	HDF5 Repository Adapter	Translates heterogeneous storage formats into the Domain's repository interface.
Strategy (GoF)	Rendering backends	Interchangeable visualization backends (Matplotlib, Plotly).
Factory Method (GoF)	`LISAViz` class methods	Encapsulates the instantiation of the facade and its dependencies.
Repository (DDD)	`IGalacticBinary Repository`	Decouples the Domain from storage.
Value Object (DDD)	Domain parameter models	Enforces immutability and equality by value, preventing side-effects.
Entity (DDD)	`GalacticBinary` model	Maintains identity across iterations regardless of parameter changes.
Iterator / Generator (GoF)	Chunk Streamer	Bounds memory to $O(\text{chunk\_size})$ by yielding sequential blocks.
Virtual Proxy (GoF)	HDF5/NumPy Memory Mapping	Lazily pages float values from disk on-demand when sliced.
Dependency Injection	`LISAViz` constructor	Promotes testability by injecting the repository dynamically.
Lazy Initialization	Module `__init__.py`	Defers heavy imports to reduce startup latency and isolate tests.

Table 4.2: Design patterns applied in the architecture and their justifications.

Chapter 5

Implementation

This chapter describes the implementation of the visualization library, following the architecture defined in Chapter 4 and the requirements established in Chapter 3. The development was carried out in Python, integrating with the output products of the `globalfit` pipeline. Following the systems engineering approach outlined in Section 1.7, this implementation focuses exclusively on Galactic Binaries to validate the core architecture. The chapter begins with an initial validation using a simple linear model, then progresses to the full Galactic Binary visualization pipeline.

5.1 Development Environment and Tools

The library is a standalone Python package developed in Python 3.10. It reads the `globalfit` pipeline's output products but imports nothing from the `globalfit` runtime, so it installs on its own through `pip` (`pip install ./lisaviz`) with a light set of open source dependencies and builds a self contained wheel (NFR-03). A test fails the suite if any heavy or version control dependency ever enters the runtime requirements, so the standalone property is enforced rather than assumed. The main dependencies are:

- **h5py / pandas:** For reading HDF5 files and PyTables formatted catalogs (Collette 2013).
- **numpy:** For numerical operations on MCMC chain arrays and as the array type passed between the ingestion and visualization layers.
- **arviz-stats:** For the MCMC convergence diagnostics (effective sample size, R-hat, Monte Carlo standard error and the integrated autocorrelation time) through its low-level array interface, which bypasses `xarray` entirely (Martin et al. 2026).
- **astropy:** For projecting equatorial sky coordinates onto the Mollweide map through its World Coordinate System machinery rather than a hand written transform (Robitaille et al., n.d.).
- **matplotlib:** As the rendering backend for static, publication quality plots.

- **plotly:** For interactive plots that support zooming, panning and hover inspection.
- **corner:** For generating corner plots from posterior samples, rendered through the Matplotlib backend (Foreman-Mackey 2016).
- **emcee (optional extra):** For the initial straight line fitting validation, with a plain random walk Metropolis sampler as a fallback when `emcee` is not available (Foreman-Mackey et al. 2013).

The code follows the package structure established in Chapter 4, with separate modules for data ingestion, domain models and rendering. The repository organization follows the standards proposed by Wilson et al. (2016) for scientific computing. Raw telemetry and HDF5 metadata are isolated in a `data/` directory and treated as read-only. The library implementation resides in the `lisaviz/` package directory, while generated visualizations are exported to `results/`. By strictly isolating data from code, the project keeps multi-gigabyte telemetry files out of the version control system, which prevents repository bloat while preserving reproducibility. Finally, to make collaboration within the DDPC easier, all environment dependencies are explicitly locked and the repository includes clear documentation to support onboarding for new astrophysics researchers (Wilson et al. 2016).

5.2 Initial Validation: Straight Line Fitting with emcee

Before the Galactic Binary data, the library's trace and corner modules were exercised on a simple known case: a straight line $y = mx + c$ sampled with `emcee` (Foreman-Mackey et al. 2013). Because the posterior here is simple and its shape known in advance, it is a clean check that the library renders the result correctly. Figures 5.1 and 5.2 show the trace and corner plots the library produced, recovering the convergence and the expected mild correlation between slope and intercept, which confirms the modules work before they are applied to the more complex Galactic Binary search.

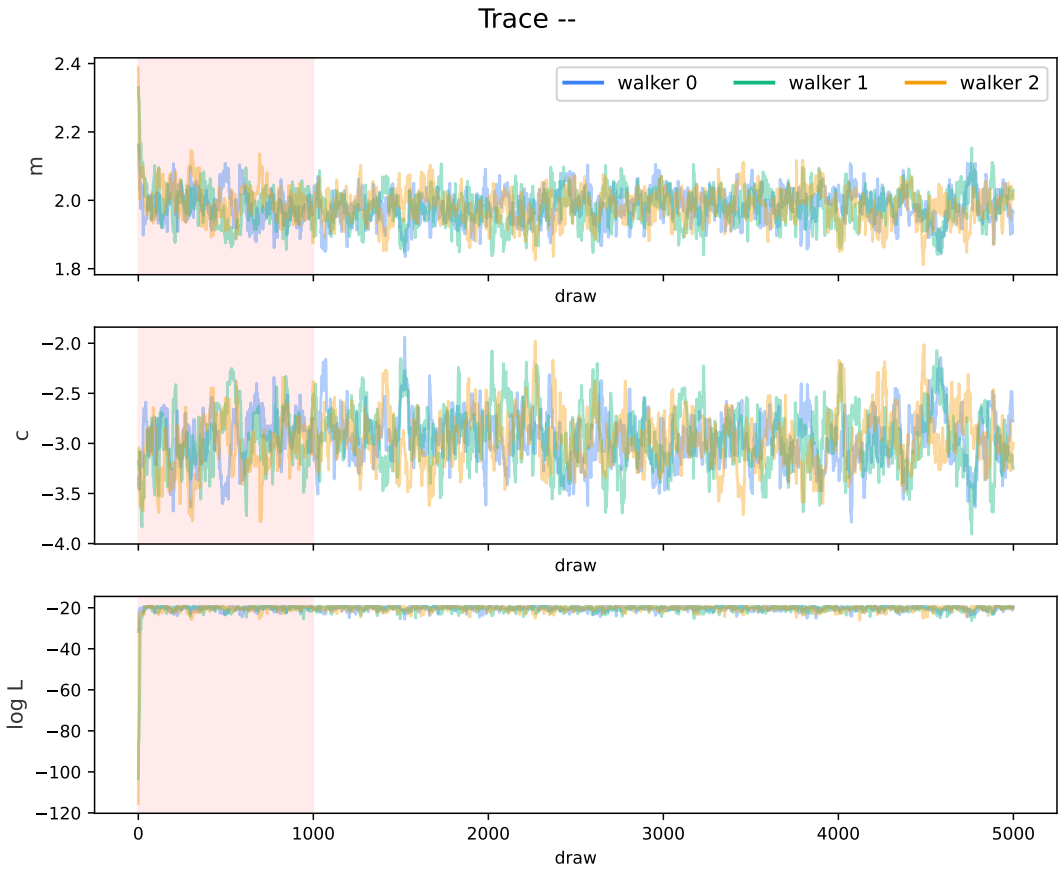

Figure 5.1: **Initial Validation Trace Plot.** Walker trajectories for the two sampled parameters (m, c) of the straight line model. Trajectories of three representative walkers (drawn from the 32-walker ensemble) converge within 200 iterations, validating the trace plot module.

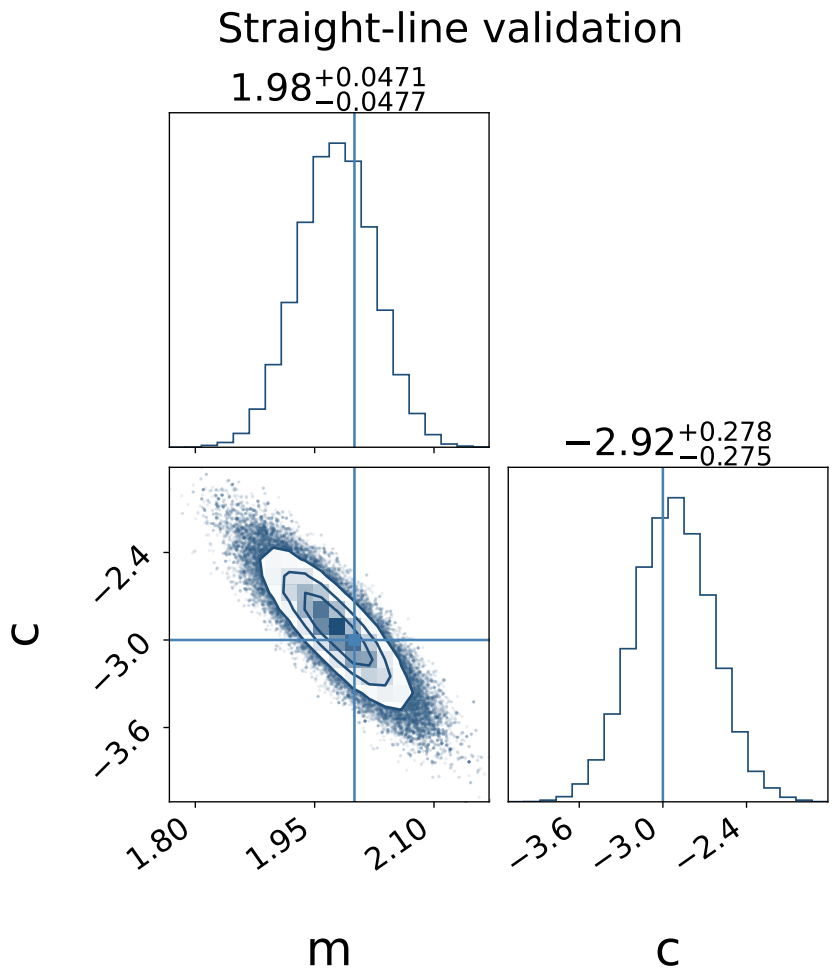

Figure 5.2: **Initial Validation Corner Plot.** Marginalized 1D and 2D posterior distributions for the straight line model. The true injected values fall within the high-density regions of the recovered distributions, confirming correctness.

5.3 Data Ingestion Layer

The data ingestion layer implements FR-01 (Table 3.1) by providing a structured interface for loading Global Fit run directories. Rather than requiring the user to manually parse HDF5 files and numpy arrays, the library exposes a single entry point that automatically discovers and deserializes all relevant data products. This design aligns with the scientific computing convention of treating raw datasets as immutable (Wilson et al. 2016). The ingestion layer reads the Global Fit run products without altering the original files on disk, feeding clean, tidy domain objects to the visualization layer.

The pipeline does not emit a single format, so the path from disk to domain object has to be made explicit before any plot is drawn. Figure 5.3 sets it out. A run directory is organised first by iteration and then by frequency sub-band. Within a sub-band the `HDF5RepositoryAdapter` reads each artifact through its own method: `central_catalog.h5` becomes a `GBCatalog` through

`get_catalog`, the flattened `new_source/chain_*.npy` parameter estimation files become an `MCMCChain` through `get_chain`, `fstats.pkl` becomes an `FstatGrid`, `noise_pars.h5` becomes a `NoiseModel` and `mbhb.h5` becomes an `MBHBCatalog`. The chain read stays bounded because `get_chain` memory-maps the `.npy` file and materializes only the requested draw range rather than the whole history. On a run whose parameter estimation stage has not written its output yet, `get_chain` falls back to the four parameter `fstats/chain_*.npy` search chains of the search stage. The chunk layout is checked once when the adapter is constructed, so a file whose layout would thrash under sequential reads is rejected at that point rather than part way through a render. Holding this mapping in one place is what keeps the rest of the library working in domain terms, because no module above the adapter ever sees an HDF5 path or a NumPy field name.

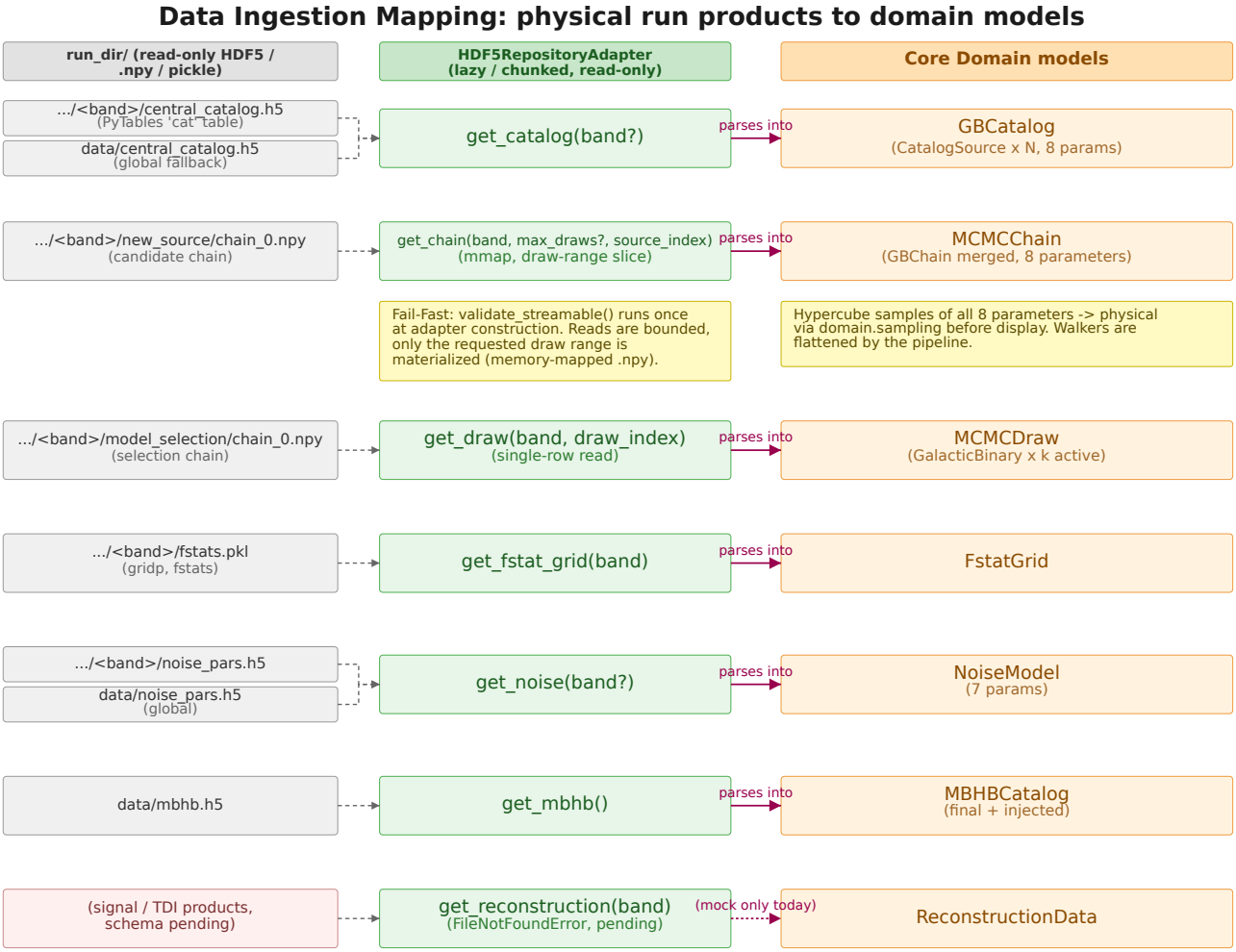

Figure 5.3: **Data ingestion mapping.** Each Global Fit run directory is organised by iteration and then by frequency sub-band. The `HDF5RepositoryAdapter` reads each on-disk artifact through a dedicated method and parses it into a single domain model, so all knowledge of the HDF5 and `.npy` layout stays inside this one layer. The chain read is bounded to the requested draw range and the chunk layout is validated once at construction.

5.3.1 Iterator Implementation

The architectural Iterator pattern of Section 4.5 is realized in Python through Generators, functions that `yield` rather than return. The realization spans the three layers as follows:

1. **Manifested as a Generator (Ingestion Layer):** The lowest-level streaming function slices the HDF5 dataset into hyperslabs of row size `block_size`. By using the `yield` keyword instead of returning a list, calling this function returns a Generator Iterator object. `yield` pauses execution, returns the chunk and waits for the next call.

2. **Propagated through the Repository Adapter:** The repository adapter (`hdf5_adapter.py`) wraps and passes this generator iterator up to the visualization layer. It iterates over the raw data blocks, packages them into domain structures and then `yield`s them upwards.

3. **Consumed Sequentially (Visualization Layer):** In the visualization layer, functions like the sky map generator consume this iterator using a standard `for` loop. Python automatically calls `__next__()` on the iterator, loading exactly one block into memory, processing it and discarding it.

This mechanism achieves strict separation of concerns because the visualization loop does not need to know how data is sliced or that it resides in an HDF5 file. Because Python garbage-collects the old chunk dictionary at the end of each loop iteration before requesting the next chunk, memory consumption remains strictly bounded to $O(\text{block\_size})$.

5.3.2 Domain Models

The ingestion layer maps raw HDF5 data into typed Python objects. The core domain models are:

- **GBCatalog:** the central catalog (`central_catalog.h5`), containing the 8 physical parameters plus metadata (SNR, Bayes Factor, confidence) for every resolved Galactic Binary.
- **GBChain:** a Galactic Binary parameter estimation chain, stored in sampling (hypercube) space exactly as written to disk. Its parallel tempered walkers are merged in the on-disk product, so it holds all 8 sampled parameters (the 4 intrinsic and 4 extrinsic) plus the log-likelihood and log-posterior across all iterations.
- **NoiseModel:** the 7 parameter noise fit (`noise_pars.h5`), including instrument noise and confusion foreground parameters.
- **MBHBCatalog:** the Massive Black Hole Binary catalog (`mbhb.h5`), including both the final parameter estimates and the pre observation injected values.

Listing 5.1 shows the core of the model. The Value Objects are immutable and compared by value, the `GalacticBinary` Entity is compared by identity through its `source_id` and the `MCMCDraw` Aggregate holds a variable length list of sources, exposing the trans-dimensional source count as `nsource`.

Listing 5.1: Core domain model: the immutable Value Objects, the `GalacticBinary` Entity and the variable length `MCMCDraw` Aggregate.

```
@dataclass(frozen=True)            # Value Object, equality by value
class IntrinsicParameters:
    f0: float                      # frequency
    fdot: float                    # frequency derivative
    declination: float
    right_ascension: float

@dataclass(frozen=True)
class ExtrinsicParameters:
    amplitude: float
    inclination: float
    polarization: float
    initial_phase: float

@dataclass(eq=False)               # Entity, identity by source_id
class GalacticBinary:
    intrinsic: IntrinsicParameters
    extrinsic: Optional[ExtrinsicParameters]  # None for an F-statistic search draw
    snr: float
    source_id: str

@dataclass(frozen=True)            # Aggregate Root
class MCMCDraw:
    iteration_index: int
    sources: list                  # variable length per draw
    walker_id: int
    log_likelihood: float

    @property
    def nsource(self):             # the trans-dimensional count, a first class
    fact
        return len(self.sources)
```

The detailed class structure of these models, including their mapping to the underlying HDF5 schema, is also documented in the domain model UML diagram (Figure 4.3).

5.4 System Lifecycle and Pipeline Flow

The library turns a run directory into rendered figures through one repeated call path. Figure 5.4 traces this pipeline, from constructing the repository to returning the final figure. The adapter validates the chunk layout once at construction to prevent thrashing. A render then runs through the `LISAViz` facade: `get_chain` reads only the requested draw range from the memory-mapped `.npy` and builds an `MCMCChain`, which the plotting function draws through the chosen `RenderBackend`.

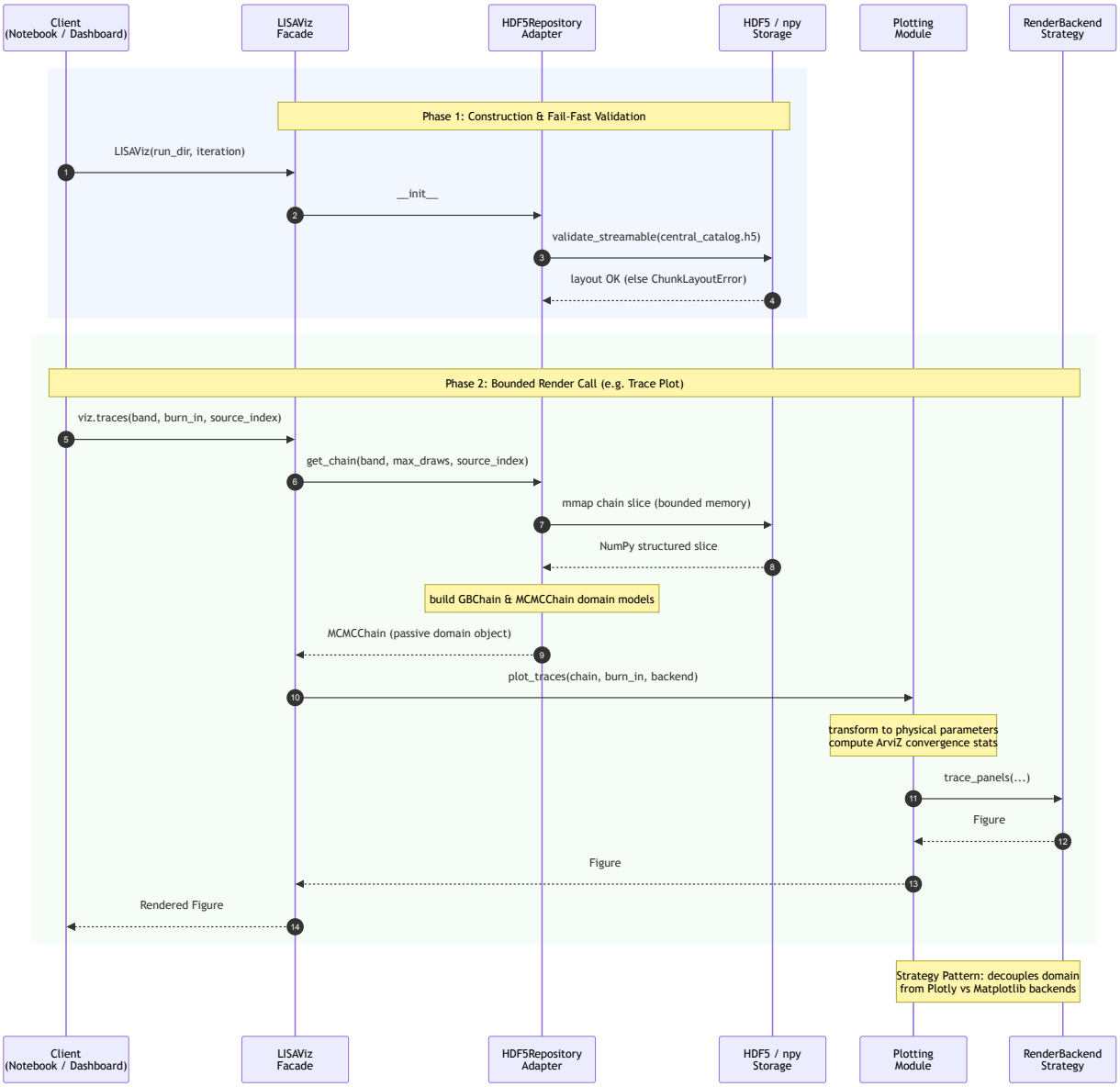

Figure 5.4: **Render call lifecycle.** The domain objects stay passive while the facade orchestrates the call and the backend Strategy decides Plotly or Matplotlib.

5.5 Galactic Binary Visualization

The Galactic Binary visualization module implements FR-02 through FR-05 and is the core of the library's diagnostic output. It provides four main plot types, each addressing a specific analytical need identified in the user scenarios (Section 3.2.1).

The figures in this section come from two sources, stated here so each is read for what it is. The sky map runs on the real resolved catalog of a local Sangria run, 13,106 sources streamed through the bounded path of Section 4.5. The corner, the trace and the convergence flag are rendered from the mock repository's synthetic chains, which carry all eight sampled parameters across several

walkers, because the local runs write only the four parameter search chains of the search stage. The autocorrelation of Figure 5.8 runs on one of those real chains, which is why it shows four parameters. All of it enters through the same repository interface, so the render path is identical whichever adapter supplies the data.

5.5.1 Interactive Sky Map (FR-02)

The sky map renders the spatial distribution of resolved Galactic Binaries in equatorial coordinates (α, δ). Each source is drawn as a point whose color encodes the gravitational wave frequency through the perceptually uniform `cividis` colormap and whose size grows with the source's uncertainty, so that the least reliable sources are the largest marks. This polarity follows the convention physicists already read from error ellipses, where a larger mark means a wider spread. It directs attention to the sources that most need scrutiny rather than to the ones already well determined. From the map the physicist can quickly read the spatial coverage and spot clusters or gaps in the source catalog.

Size is driven by the per source confidence score carried in the catalog, which is not a raw Signal-to-Noise Ratio (SNR). The pipeline derives this score in stages. A sigmoid of the SNR sets a best case value in $[0, 1]$. For a source already seen in a previous iteration this value is replaced by the cross iteration waveform overlap, so that the source counts as confident only if the sampler recovers the same waveform from one iteration to the next. Finally a penalty proportional to the cumulative overlap with neighbouring sources is subtracted, so that a loud but confused source is pushed down before the result is floored at zero. The mark size is taken as the complement of this score, which means a source that is faint, unstable between iterations or entangled with its neighbours appears large on the map. Because the confusion penalty is built into the score, the regions where the Galactic foreground is most blended, which is the central difficulty of the Global Fit, are exactly the regions the sky map enlarges.

Figure 5.5 shows this view in its streaming form, run on the real resolved catalog of a local run. The reduction behind it is set out below.

The equatorial coordinates are projected onto the Mollweide map through astropy's World Coordinate System machinery rather than a hand written transform, which keeps the projection consistent with the conventions astronomical sky surveys use. The Mollweide projection preserves area and is standard in those surveys (Snyder 1987). Density reduction is inspired by the Splatterplots approach of Chapter 2 along two paths, keeping its principle of preserving outliers while abstracting dense regions but replacing its kernel density contours with hexagonal binning. The in-memory path draws every point until the catalog passes roughly 5000 sources, then thins the scatter, keeping isolated, most uncertain and flagged sources while dropping the dense interior. The streaming path uses no threshold, always binning the catalog into a hexagonal density field while keeping those same outliers as points, so the view stays legible regardless of count and is built to scale toward the Mojito field.

The reservoir is a fixed-capacity selection keyed on $1 - \text{confidence}$: it retains the k most-uncertain sources seen so far. An arriving source bumps the least uncertain one kept whenever it

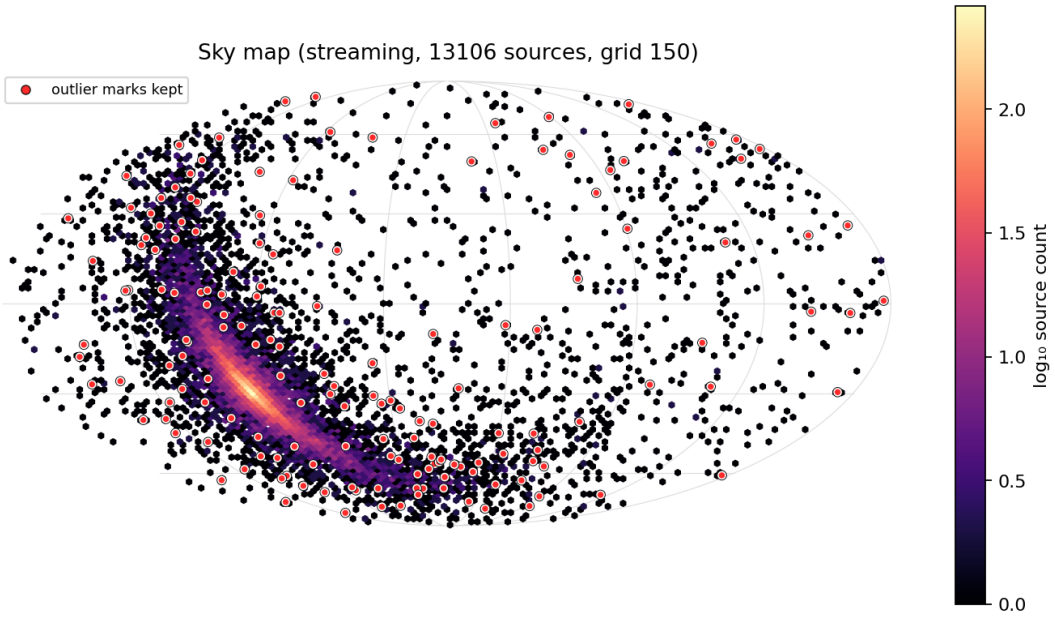

Figure 5.5: **Streaming sky map of the resolved Galactic Binaries from a local Sangria run.** The 13,106 sources of the real resolved catalog, streamed in fixed blocks into a Mollweide density field whose hexagonal cells are coloured by $\log_{10}$ source count, so the Galactic disk and bulge emerge straight from the data. The bounded reservoir keeps the most uncertain sources as red marks sized by $1 - \text{confidence}$, so the sources that most need scrutiny stay visible above the reduction.

is more uncertain, so the points left at the end are the most uncertain in the whole field, not the first the stream delivered. Arrival order cannot change the result. An evicted source is not lost, since it drops into the density field with the rest of the interior. The point layer omits only the least uncertain sources, which is what it should omit, since the points exist to flag the ones worth scrutiny.

When reduced this way each cell is coloured by its source count on a log scale rather than by frequency, since the count is what the density reduction recovers and it ranges over several orders of magnitude, from a few sources in a sparse cell to hundreds of thousands in the Galactic bulge, which a linear scale would flatten. The colorbar is labelled in $\log_{10}$, so the scale is declared rather than hidden and the perceptually uniform map then makes equal colour steps read as equal factors in count, the same convention as a logarithmic axis. Frequency is kept per cell and the reduction is held per frequency sub-band, so the overview can still be filtered or split by frequency rather than losing it. The waterfall is the same reduction in another frame, so both give a future dashboard a per sub-band filter on one bounded memory budget.

The population view also includes a waterfall companion to the sky map: a scatter of each resolved source by gravitational wave frequency against its amplitude or signal to noise ratio, coloured by frequency and sized by the same $1 - \text{confidence}$ rule. Where the sky map shows where the sources sit, the waterfall shows how they spread across the band, which makes it easy to see where the sampler is finding sources and where the confusion foreground prevents resolution.

5.5.2 Corner Plot (FR-03)

The corner plot module generates N dimensional marginalized posterior plots from the MCMC chains. The module is not fixed to a parameter count. For a Galactic Binary the grid is 8×8 (Section 2.1.3).

Figure 5.6 shows the result for one source.

The implementation handles the coordinate transformations required to convert from the MCMC sampling space (e.g., the folded hypercube frequency and $\sin\delta$) to the physical parameter space (e.g., f_0 and δ) before rendering, so the displayed posteriors are in physically meaningful units.

5.5.3 Trace Plot (FR-04)

The trace plot module visualizes the evolution of individual walker trajectories across MCMC iterations. This diagnoses convergence. It renders one panel per sampled parameter, a log-likelihood panel and, when a trans-dimensional source count is present, an `nsource` panel that tracks how it changes across the run. Flagged walkers are drawn in red and the burn-in region is shaded. Alongside the traces the module reports the per parameter diagnostics computed through the arviz-stats array interface, the effective sample size, R-hat, the Monte Carlo standard error and the integrated autocorrelation time, with a companion autocorrelation plot (Figure 5.8) that annotates each parameter with its τ and the acceptance fraction.

The R-hat statistic needs care because the parameter estimation chain is written flattened, with its parallel tempered walkers already merged into one array. The library reads just this flattened chain for now, so it computes a split R-hat over the two halves, the standard single chain form of the diagnostic. It labels the value with the form that produced it, so a split result is not read as a between walker one. A split R-hat reports only within chain stationarity, whether the run has settled and its two halves agree, which a chain trapped in a single mode would still satisfy while missing the others. The pipeline's own criterion is a between walker R-hat below about 1.2 across the parallel tempered walkers (Deng et al. 2025), which needs the separate per-walker chains the library does not read yet. Reading them for that stronger diagnostic is a direct adjustment rather than a redesign, because the adapter already loads each chain file as one walker, so the same arviz-stats interface computes the between walker form the moment several are supplied.

Figure 5.7 shows the trace for one Galactic Binary chain.

5.5.4 Possible Non-Convergence Flag (FR-05)

To address the need to find chains that may not have converged or got stuck in a local minimum (US-02), the library computes its own heuristic, since the pipeline writes no convergence verdict. For each frequency band it takes the mean log likelihood of the final 20% of the chain, then flags bands that fall anomalously low against a robust median and median absolute deviation threshold over the neighbouring bands. The median and median absolute deviation are used rather than the mean and standard deviation because a few badly fitting bands would inflate the latter and mask the outliers the flag is meant to catch. This catches a failure R-hat cannot: a chain that mixes

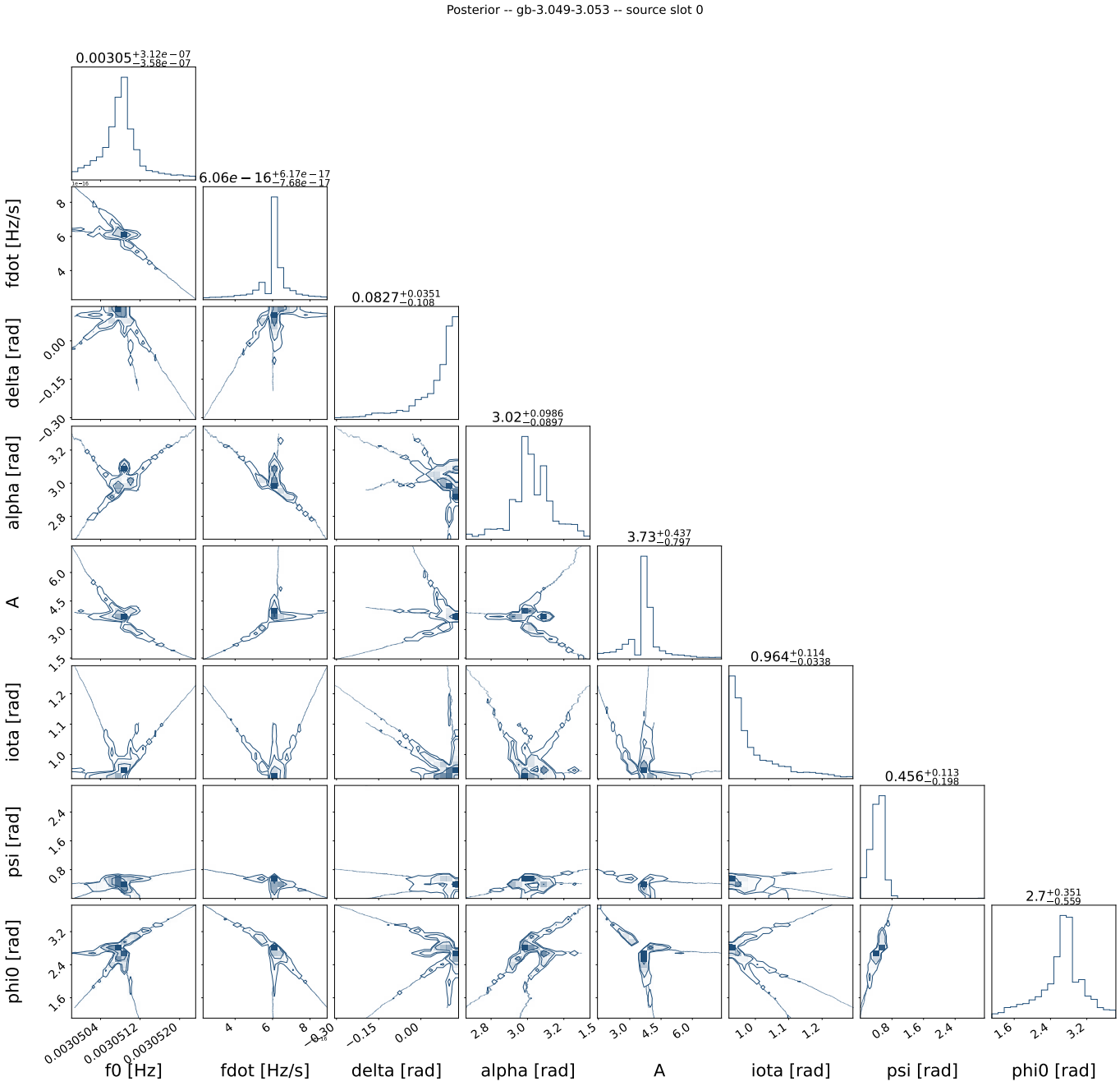

Figure 5.6: **Corner plot of a single Galactic Binary.** The 8×8 grid of the eight sampled parameters, with the 1D marginal posteriors on the diagonal and the 2D contours below it.

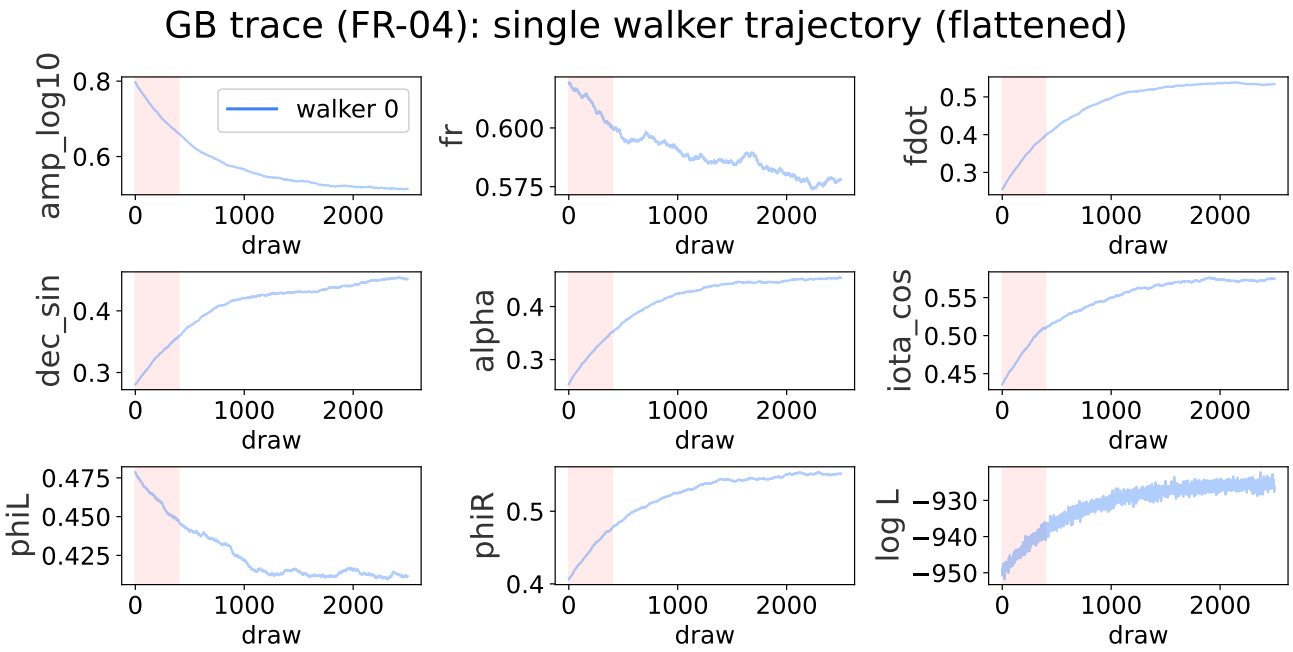

Figure 5.7: **Trace plot of a Galactic Binary chain.** Walker trajectories for the eight sampled parameters across iterations, on a synthetic chain built so converged walkers sit beside a stuck one. The shaded region marks the discarded burn-in.

cleanly can still settle in a low local minimum, where R-hat sits near 1 yet the band still reads as a likelihood outlier. A genuinely faint band reads low too, so the flag raises a candidate rather than a verdict. A flagged chain shows up as a highlighted band on the sky map, which the physicist can open through the trace plot module.

Figure 5.9 shows the bands the heuristic raises.

5.6 Signal Reconstruction (FR-06)

The Power Spectral Density (PSD) residual plot compares the raw observed data against the sum of all reconstructed source models and the fitted noise curve. This visualization directly addresses US-04 by allowing the physicist to identify frequency regions where the model does not adequately explain the data, indicating either a missed source or a poor noise fit. A fractional residual panel sits below the overlay and, where the noise model is available across iterations, the module overlays the noise PSD as it evolves.

The same module also overlays recovered waveforms on the observed TDI spectrum. When an injection and a noise PSD are supplied, each recovered source is annotated with its match score, the normalized noise weighted overlap in $[0, 1]$ that measures how closely the reconstruction reproduces the injected signal, where a self match is exactly 1. The match builds on the standard noise weighted inner product, so the comparison is weighted by the instrument sensitivity rather than treating every frequency equally.

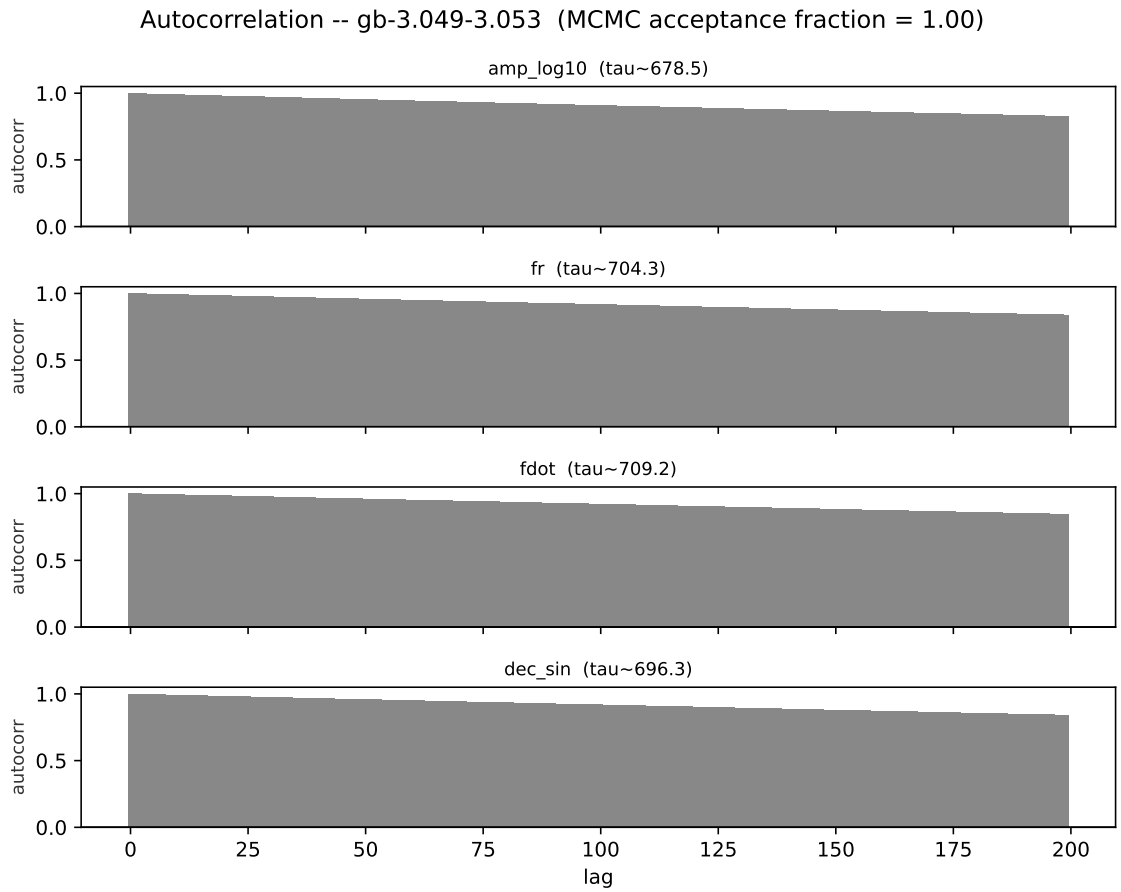

Figure 5.8: **Autocorrelation diagnostic for one Galactic Binary sub-band.** The autocorrelation of each of the four intrinsic parameters against the lag, annotated with its integrated autocorrelation time τ and the MCMC acceptance fraction. Larger τ means slower decorrelation and fewer independent draws.

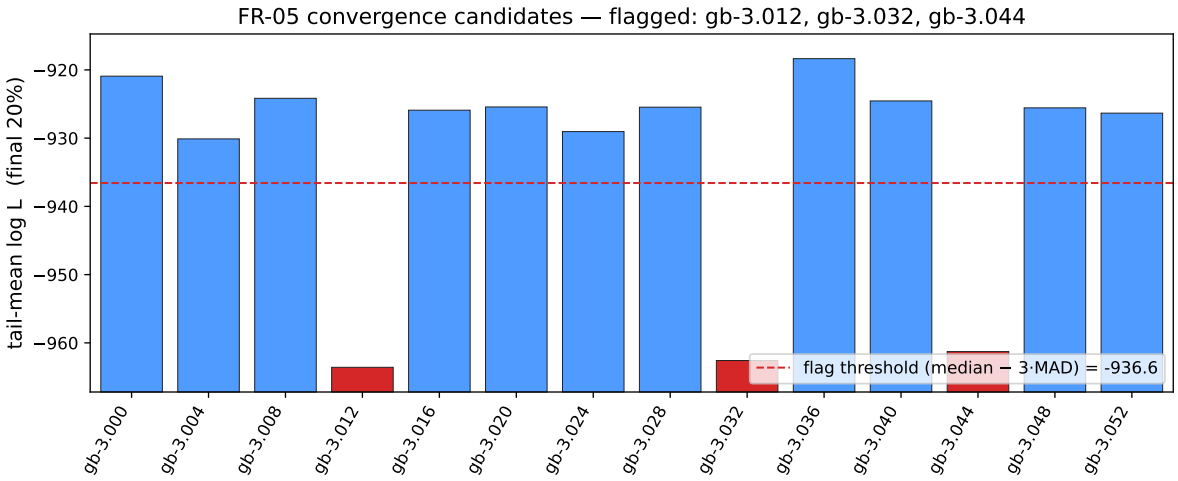

Figure 5.9: **Convergence candidates flagged by the FR-05 heuristic.** Frequency bands whose mean log-likelihood over the final fifth of the chain falls anomalously low against a robust median and median absolute deviation threshold are highlighted for inspection rather than certified as failed.

Figure 5.10 shows this comparison on a constructed spectrum whose noise and source content are known, so the view is read against a known answer.

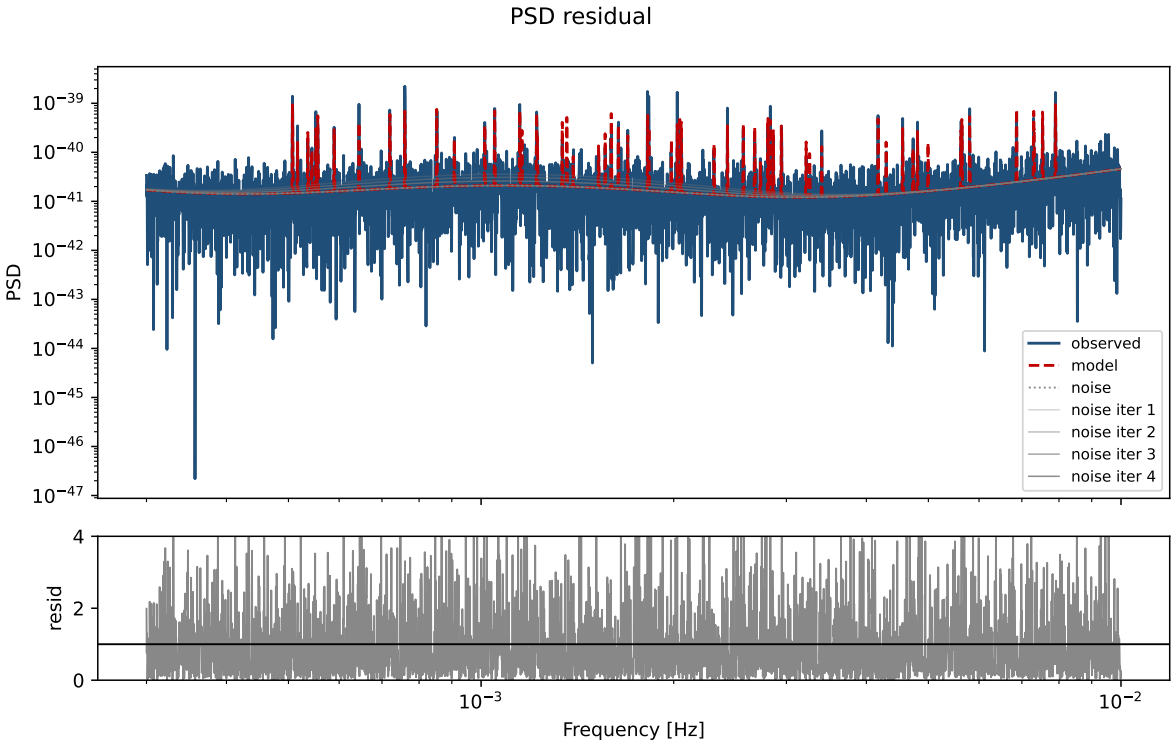

Figure 5.10: **Power spectral density residual.** The observed data spectrum against the sum of the reconstructed source models and the fitted noise curve, with the fractional residual below. Excess power marks a frequency region the model does not yet explain.

Figure 5.11 shows the overlay with the per source match scores on the same constructed data.

5.7 F-statistic Landscape (FR-09)

The F-statistic grid that seeds the search (Section 2.1.3) is itself a useful diagnostic surface. FR-09 reads that grid (from `fstats.pkl`) and renders it as a 2D contour surface over a pair of intrinsic parameters, which is what US-05 asks for: seeing the parameter degeneracies before launching an expensive sampling run.

The grid covers all four intrinsic axes at once, so two of them are chosen for the plane and the rest are reduced. Each cell shows the profiled value, the maximum F-statistic over the axes that are not displayed, rather than a slice at one fixed value. Profiling is what keeps a degeneracy ridge visible: when two parameters trade off, the ridge of high F-statistic survives the projection instead of being washed out. The two displayed axes are mapped from the sampling hypercube to physical units through the same transform the corner plot uses, so the contour reads in physical units.

Figure 5.12 shows the landscape for one band.

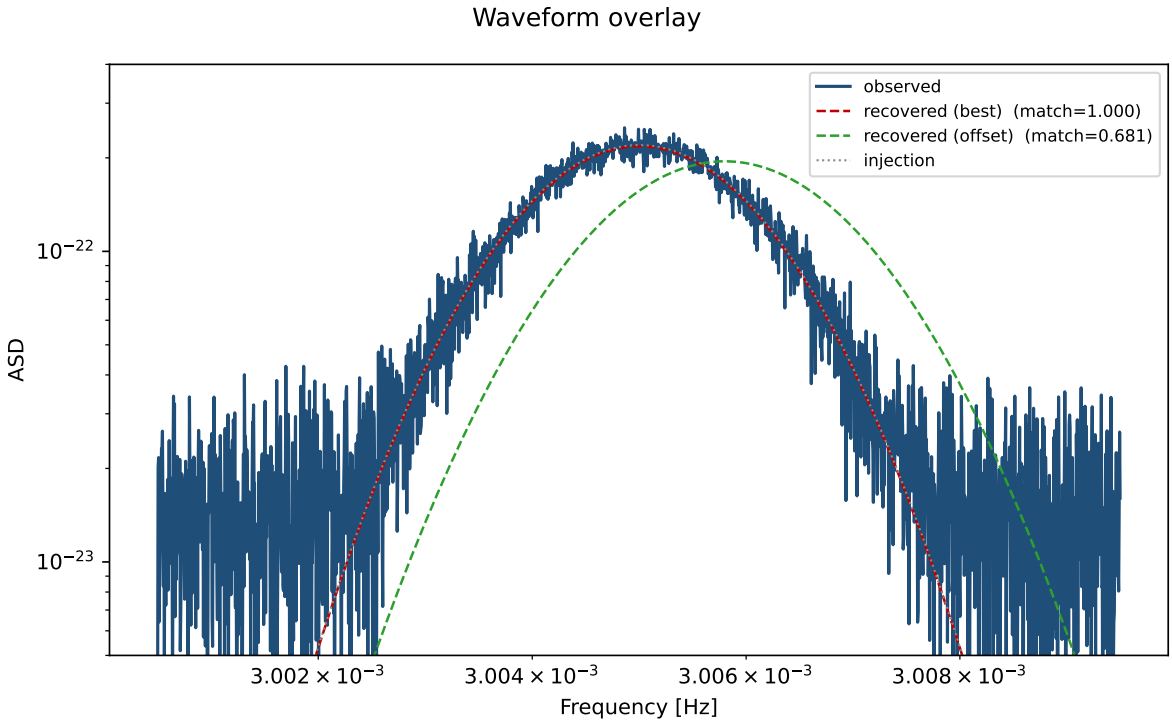

Figure 5.11: **Frequency domain waveform overlay.** The amplitude spectral density of the observed TDI data with the recovered source waveforms drawn over it. Where an injection and a noise model are available, each recovered source carries its match score, the noise weighted overlap in $[0, 1]$.

On a real Global Fit F-statistic grid the four axes are sampled at different resolutions, frequency $\times$ frequency derivative $\times$ $\sin\delta$ $\times$ right ascension at $100 \times 1 \times 10 \times 20$, with the frequency derivative held fixed. The library defaults to the two best resolved axes, frequency and right ascension, excluding the fixed derivative axis rather than drawing a degenerate single cell row. Figure 5.12 shows a different pair, the frequency against declination plane for one band, which is where this grid shows its clearest degeneracy ridge. Run against this grid of 20,000 evaluations the contour surfaces the grid maximum, an F-statistic of 96.4, instead of flattening the peak when it reduces the hidden axes.

Adding this view followed the extension path the architecture defines, the same path a later source type like an MBHB or EMRI would take. It needed a new domain model (`FstatGrid`), one method on the repository interface (`get_fstat_grid`), one reader in the HDF5 adapter, a synthetic equivalent in the mock, a contour method on each rendering backend and its tests. No existing module's internals were touched.

5.8 Dashboard Layout

The library's plotting modules compose into a single monitoring interface. Before describing the physical layout, Figure 5.13 maps the user interaction workflow. Letting the analyst navigate from

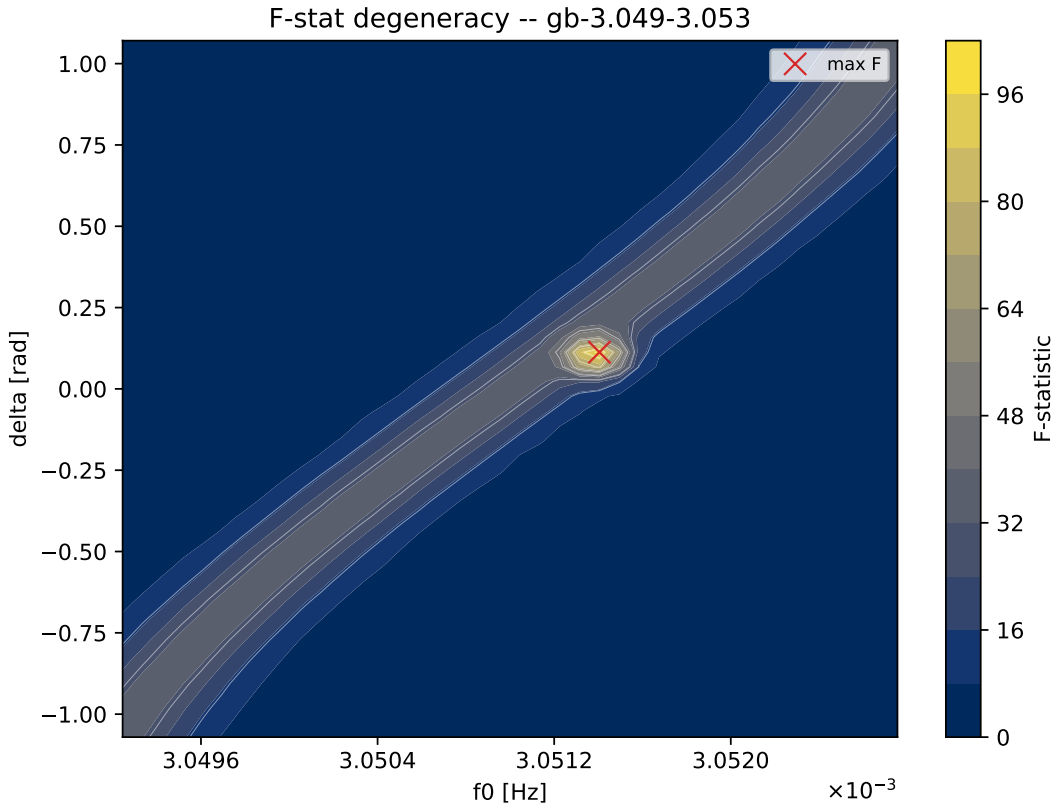

Figure 5.12: **F-statistic landscape (FR-09).** A 2D contour of the F-statistic over a pair of intrinsic parameters, each cell the profiled maximum over the axes not shown, so a degeneracy ridge survives the projection. The displayed axes are mapped to physical units.

macroscopic population overviews directly into microscopic single-source diagnostics directly addresses Objective 3.

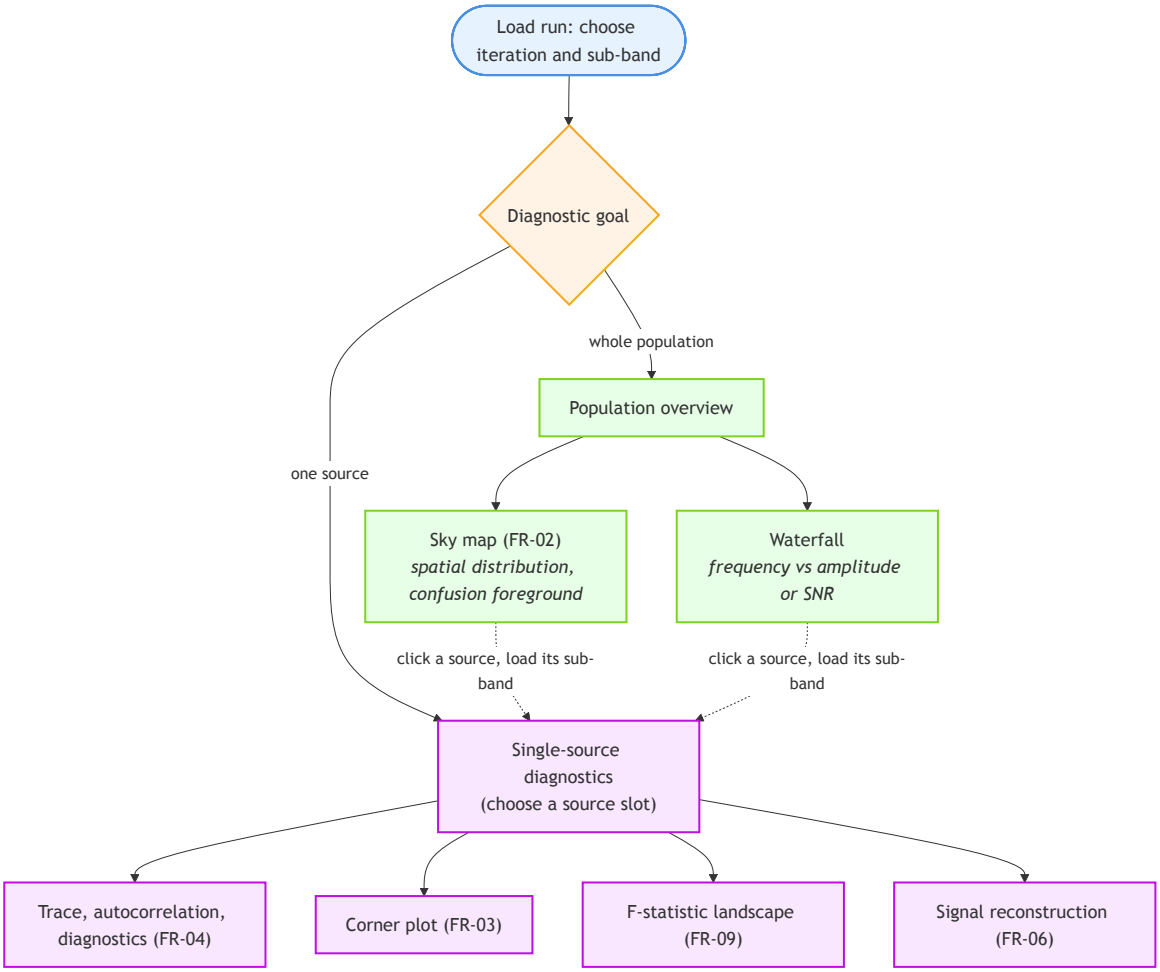

Figure 5.13: **User Interaction Workflow.** The decision tree mapping the diagnostic paths available to the user.

Figure 5.14 proposes a structural layout built on the perceptual rules of Chapter 2, dividing the interface into a top filter bar and a two-column main canvas to avoid the overflow pagination that breaks visual continuity (Bach et al. 2023).

Zone 1: Global Filters. A horizontal bar spanning the top of the interface controls the data source, active frequency sub-band (FR-08) and confidence thresholds. It uses a distinct preattentive color to capture initial attention (Few 2006), while placing filters at the top preserves the entire width of the main canvas below for charting real estate.

Left Column: Astrophysics and Inference. Following Shneiderman's hierarchy (Lin et al. 2024), the left column splits vertically. The top half is the entry point, anchoring the Sky Map (FR-02) and Waterfall overviews strictly in the top-central-left position to minimize cognitive strain (Zhang et al. 2024). The bottom half holds detailed multi-field views: the 8x8 Corner Plot (FR-03),

F-statistic degeneracies (FR-09), Waveform Overlay and PSD Residual (FR-06). Clicking a source in the overview drives these bottom panels via Coordinated Multiple Views (Lin et al. 2024), providing detail on demand.

Right Column: Sampler Health. The right column is a tabbed container dedicated to MCMC diagnostics: Trace Plots (FR-04), Autocorrelation and tabular summaries. Because the sampler produces traces for eight parameters alongside likelihood and source counts, they require substantial vertical space. Within their respective tabs, the Trace and Autocorrelation views render their plots using the Small Multiples pattern introduced in Chapter 2. The trace panels share a common draw number on their X-axis, while the autocorrelation curves share the lag. By stacking these vertically at the same width and scale, the full parameter space can be compared directly rather than from memory (Lin et al. 2024).

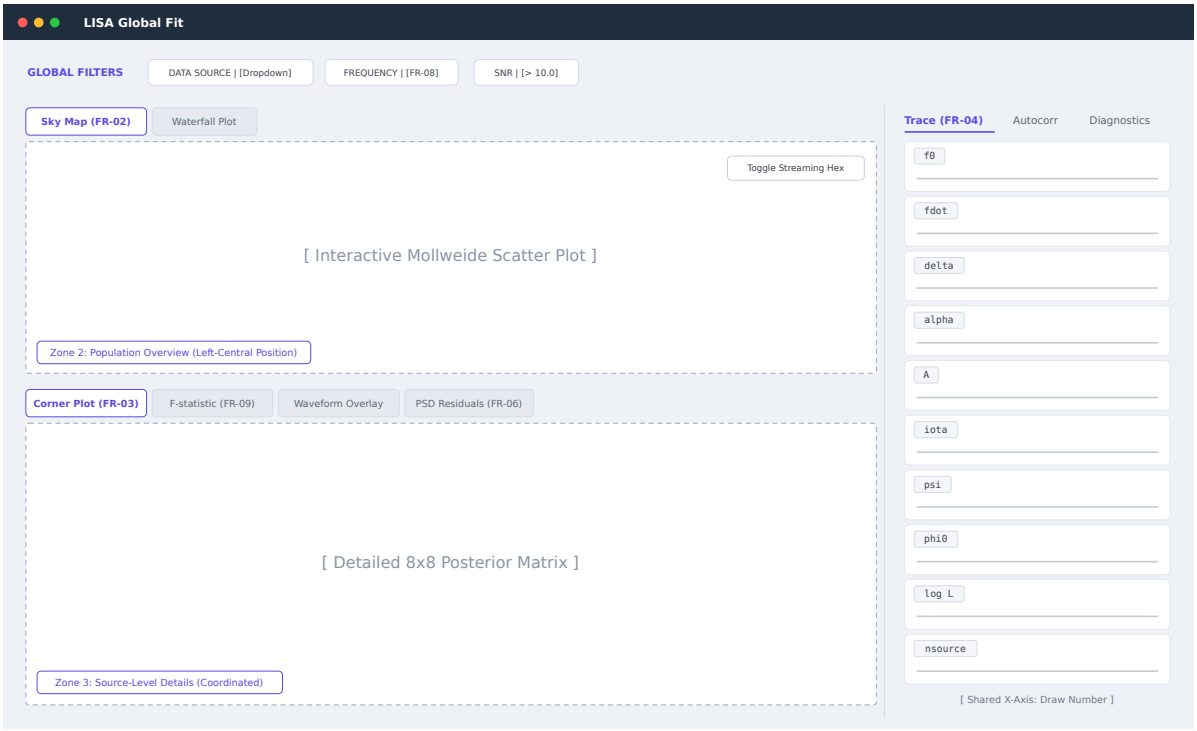

Figure 5.14: **Proposed dashboard layout.** The interface uses a top bar and a 2-column main canvas grounded in cognitive load principles. A horizontal bar holds global filters. The left column splits vertically between the population overview (top) and source-level details (bottom). The right column dedicates the full screen height to MCMC health diagnostics, stacking trace plots as small multiples.

Each diagnostic view also carries an on-demand help panel, collapsed by default so it never crowds the chart. It applies the cheat sheet idea of Wang et al. (2020), but carries only the two families that earn their place for an expert reader, the pitfalls and the false friends, each tied to a misreading this tool's own rendering can introduce. Three entries are carried. On the population views a larger mark means a less certain source rather than a stronger one, since the size encodes one minus confidence. The density field there is a count of sources rather than a posterior surface,

so a dense blob marks where many sources sit rather than the probable place of one. Across both, a convergence flag marks a candidate to inspect rather than a verdict that the chain failed to converge. The text lives in the library beside the plotting functions rather than in any one interface, so a notebook user reaches the same guidance and a dashboard view would only decide when to surface it.

This layout is a design rather than a built deployment. It shows that the library's decoupled modules compose into a unified interface under established layout principles, the foundation a future web based monitoring dashboard would build on (Basset 2024). Every view proposed in this section is already implementable via simple function calls to the `lisaviz` API. In practice, while this rigid layout serves general analysis, the underlying library components remain independent, allowing a future deployment to introduce the drag-and-drop flexibility that experts require to adapt the environment to their specific workflow (Vuckovic, Wolosiuk, and Schmidt 2024).

Chapter 6

Evaluation

This chapter evaluates the visualization library against the objectives defined in Section 2.5 and the formal requirements specified in Section 3.2. The evaluation follows the methodology outlined in Chapter 3, assessing both functional compliance through a traceability matrix and non functional performance through empirical testing.

6.1 Requirements Verification

To verify that the artifact fulfills its intended purpose as a scientific diagnostic tool, each requirement was checked against the implemented library. Table 6.1 presents the compliance status for all functional and non functional requirements.

6.2 Quality Measures

The quality requirements of Table 4.1 were stated as ISO/IEC 25023 measures against the ISO/IEC 25010 characteristics they serve. ISO/IEC 25023 lets a project define measures suited to its own product as long as each one maps to a quality characteristic (ISO/IEC 2016), so the evaluation uses the architectural constraints as those measures, each read from the package itself rather than claimed.

Two of them are measured by the test suite. The Core Domain holds zero imports from infrastructure libraries (Reusability). This is not checked by reading the code. An architecture test parses every module under `domain/` and fails the suite if it sees `h5py`, a plotting backend, `arviz`, `astropy`, `pandas` or `xarray`, so the count stays at zero by construction of the build rather than by hand. The domain and visualization layers are also exercised without a single real HDF5 file (Testability), because the test suite runs against an in-memory mock repository that produces synthetic domain objects with no filesystem present. The full test suite passes.

The Modifiability measure is satisfied by the same boundary. A change in the upstream format, such as the L2-to-L3 move to `snake_case` column names and batched files, reaches only the one adapter that implements the repository interface. The Core Domain and the visualization layer need

ID	Requirement	Status
FR-01	Parse multi-format run directories into domain models	**Verified** (Sec 5.3)
FR-02	Render interactive sky maps	**Verified** (Sec 5.5)
FR-03	Render N-dimensional corner plots	**Verified** (Sec 5.5)
FR-04	Render individual walker trace plots	**Verified** (Sec 5.5)
FR-05	Flag chains as candidates for possible non-convergence	**Verified** (Sec 5.5)
FR-06	Render PSD residual plots	**Verified** (Sec 5.6)
FR-07	Export publication-quality static images	**Verified** (static Matplotlib backend, Sec 5.5)
FR-08	Filter visualizations by frequency subband	**Verified** (per sub-band reads, Sec 5.3)
FR-09	Render 2D F-statistic contour surfaces	**Verified** (Sec 5.7)
NFR-01	Maintain strict decoupling (layered architecture)	**Verified** (architecture test, Sec 6.2)
NFR-02	Strictly bound RAM during rendering	**Verified** (streaming population views, Sec 6.3)
NFR-03	Installable via standard Python `pip`	**Verified** (packaging test, Sec 5.1)
NFR-04	Absorb upstream format change in one adapter	**Verified** (adapter boundary, Sec 6.2)
NFR-05	Load run data on demand rather than in full	**Verified** (block and memory-mapped reads, Sec 6.3)

Table 6.1: **Requirements Compliance Chart.** Every requirement is verified, with each verdict pointing to the section that carries its evidence. Timed benchmarks of ingestion and interaction remain future measurements beyond these requirements (Section 6.3).

zero changes, because neither imports the format. This is the architectural guarantee of NFR-04 rather than a figure from a migration that has already run.

The Performance Efficiency measure, peak resident memory, is measured in Section 6.3. The streaming render path holds peak memory flat at about 322 MiB for the sky map (278 MiB for the waterfall), from one thousand synthetic sources up to the 26 million of the real Sangria injection catalog, while the object materializing path grows with the source count to about four gigabytes at one million. The Interaction Capability measure, an interactive frame rate at mission scale, rests on the density reduction of Section 6.3 and is left to the rendering benchmark deferred there. Reporting the measures this way is what the standard asks for: the decoupling and testability claims rest on a count the build enforces rather than on a claim that the layers look well separated.

6.3 Performance Evaluation

The library has to handle the large data volumes typical of the LISA Global Fit (NFR-02 and NFR-05). The work here draws on a Global Fit run over the LDC2a Sangria dataset, executed locally on a standard workstation (16GB RAM, 8 core CPU) to match the analysis environment a physicist would use rather than the distributed HPC centres that produce a full run. The run wrote a `gfrun` directory whose resolved catalogs hold on the order of 11,000 to 13,000 Galactic Binaries. Sangria carries known injected sources, so these outputs are what check the library for correctness. Behaviour at full mission scale is then confirmed by streaming the real 26-million-source Sangria injection catalog through the same path (Table 6.3).

6.3.1 Memory Behaviour and Ingestion

The library reads the resolved source catalog through two paths whose memory behaviour differs in a way that is the whole substance of NFR-02. The first, `get_catalog()`, is the convenient one: it reads the catalog and builds the full set of typed domain objects, four Python objects for every source, so a caller receives the entire resolved population in memory and can inspect any part of it at full fidelity. Because it keeps one set of objects per source, the memory it uses grows with the number of sources. The second, `sky_map_streaming()`, never holds the whole population at once. The repository streams the catalog in fixed-size blocks through `h5py` row hyperslabs, which the view folds one at a time into a fixed-size Mollweide density grid together with a bounded reservoir that keeps the most uncertain sources as discrete points and folds the rest into the grid (Section 5.5), discarding each block before the next is read, so the memory it holds at any moment is set by the block, the grid and the reservoir rather than by the size of the catalog.

Table 6.2 reports the peak resident memory of each path, measured in an isolated process on synthetic catalogs that grow from one thousand sources to one million. The two paths trade the same cost in opposite ways. The object path begins almost free, since a thousand sources are a thousand small allocations and little else, but every source it retains adds to the total, so its memory climbs steadily until a million sources reach about four gigabytes and the 15.5 million of a full Mojito field would run a workstation out of memory long before they were all loaded. The streaming path begins instead from a higher floor, because before it reads a single source it has already paid for the full resolution density grid, the bounded reservoir and the plotting and astronomy libraries it needs in order to draw, which together sit at around 322 megabytes for the sky map and 278 for the waterfall. What it never does is build on that floor as the catalog grows, since each block is binned and then thrown away, so the figure it pays for a thousand sources is the same figure it pays for the 26 million of the real Sangria injection. The waterfall is bounded by the same reduction, so the table shows both population views each holding its own flat floor while the object path climbs past them. That floor is a fixed cost of the render rather than a constant of nature, so its exact value shifts a little between runs with the timing of memory allocation, while the property the requirement turns on, that it does not move with the catalog, holds exactly in every run. The two costs therefore cross at around forty thousand sources, below which the object path is genuinely the lighter choice and above which the streaming path is the only one that stays within the reach of an ordinary machine. The chain path is bounded for the same underlying reason, because `get_chain` and `get_draw` memory-map the `.npy` files and read only the draw range that was asked for rather than the whole chain, which is the on-demand read NFR-05 asks of the ingestion layer.

This is what lets NFR-02 stand as a measured result for the streaming population views and the chain path rather than as a design argument alone. The object materializing read is kept beside them rather than removed, because it remains the natural choice for the small catalogs a physicist reads interactively, where its low starting cost matters and its growth has not yet begun to bite, while the streaming path is what carries the same population view to the confusion limited fields of the full mission, where the object path would have exhausted the machine long before the field was drawn.

Sources	Object (MiB)	Sky map (MiB)	Waterfall (MiB)
1,000	160	321	277
10,000	206	322	278
100,000	545	326	282
1,000,000	3,960	326	282

Table 6.2: **Peak resident memory by catalog size.** Peak resident memory of an isolated process for the object materializing path (`get_catalog`) against the two streaming population views, on synthetic catalogs read in blocks of fifty thousand rows. The object path grows with the source count, reaching about four gigabytes at one million sources, while the sky map stays near 322 MiB and the waterfall near 278 MiB, both flat because their peak is set by the block size rather than the source count.

The synthetic curve is built to a fixed grid and reservoir, so its flatness could be an artifact of how the catalog was generated. The test that matters is a real catalog at scale. The LDC2a Sangria training file holds the injected Galactic Binary truth, 26 million detached sources and 3 million interacting ones, larger than the 15.5 million the Mojito field targets (LISA DDPC 2026a). Streaming it through the same path gives Table 6.3: peak memory stays flat to within 1 MiB from one hundred thousand real sources to 26 million. The bound holds on real data, not only on inputs shaped to satisfy it.

Catalog	Sources	Streaming sky peak (MiB)
Resolved GB catalog (real)	11,190	322.5
Resolved GB catalog (real)	13,106	322.4
Sangria injection, detached	100,000	322.3
Sangria injection, detached	1,000,000	322.7
Sangria injection, detached	5,000,000	323.0
Sangria injection, detached	26,000,000	323.4
Sangria injection, interacting	3,000,000	322.8

Table 6.3: **Streaming peak memory on real catalogs.** The streaming sky map path on the real resolved catalogs and the injected truth catalogs of the LDC2a Sangria file. Peak memory stays flat to within 1 MiB from one hundred thousand to 26 million real sources.

A benchmark of ingestion time against catalog size, together with the rendering framerate during interaction, would round out the picture with a timing to set beside the memory. Those measurements are left to the maturing library, since the property NFR-02 turns on is the memory behaviour, which is the part settled here.

6.3.2 Rendering and Visual Overdraw

The second concern is the rendering layer, in particular the interactive Sky Map and the N dimensional corner plots, when a mission scale source field is drawn at once. A raw `matplotlib` scatter fails here for the reason set out in Chapter 2: past saturation the ink stops tracking density, so a

genuinely dense confusion region and a merely crowded one read alike. The Sky Map answers this with the Splatterplots-inspired density reduction of Section 4.5, hexagonal binning over the interior with statistically isolated sources kept as discrete points. What this section adds is the behaviour on data. The abstraction is exercised on large synthetic catalogs and on the real 26 million source Sangria field of Table 6.3, where above a five thousand source threshold the dense regions collapse into the hexagonal density field instead of thousands of individual glyphs, so the view stays bounded in memory as the field grows. A timed rendering benchmark of the full Sangria and Mojito fields at interactive frame rates is the measurement left for later.

Figure 6.1 shows this on a rendered field, the empirical counterpart to the conceptual Figure 2.8.

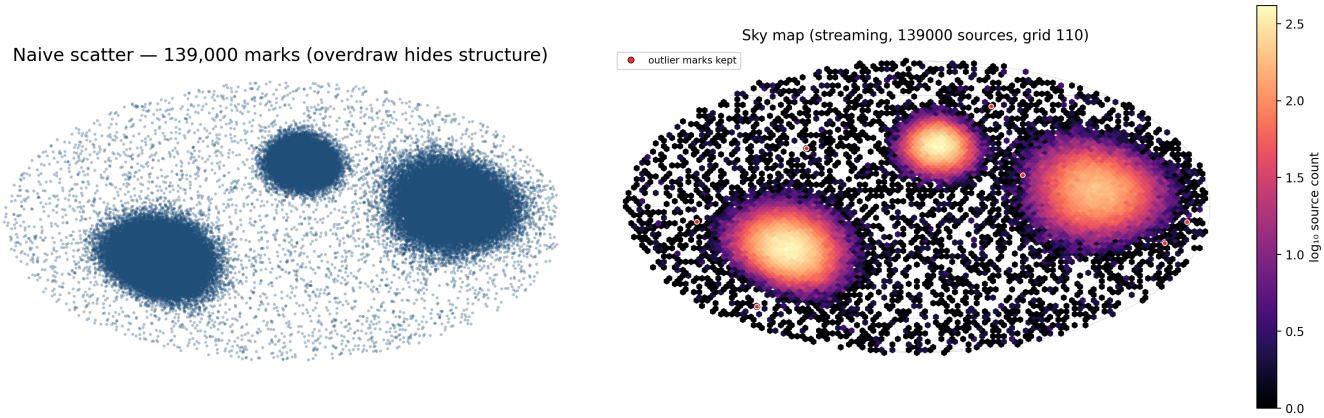

Figure 6.1: **Overdraw against density reduction.** The same synthetic source field twice. The naive scatter on the left saturates, so the dense clusters read as flat ink and their internal structure is lost. The right panel is the library's own streaming sky map render of that field, the same code path behind Figure 5.5, whose $\log_{10}$ count cells recover the density gradient inside each cluster while the isolated sources stay visible as discrete marks.

6.4 Heuristic Evaluation

The primary evaluation rests on requirements traceability. The interface was additionally inspected against established visualization heuristics, a method suited to specialized tools where a large user pool is not yet available (Paz and Pow-Sang 2016). This inspection was run by one evaluator against fixed principles, so it checks the figures for conformance to those principles rather than standing in for a usability study with the physicists. That user facing validation is the behavioural method left as future work at the end of this section. The inspection follows the computer assisted method of Zhu and Gumieniak (2021), which reads the plot JSON the interactive Plotly views emit and sorts the heuristic rules into three classes: those a program can check, those it can only flag as advice for a human and those too abstract for a program at all. The automated function `heuristic_eval` runs the first class, verifying that every plot carries a title, every cartesian plot labels both axes, every colour scale is perceptually uniform rather than a rainbow and no raw scatter exceeds the density reduction threshold (defined in Chapter 2 and applied in Section 5.5). Running it over the

seven interactive figures gives Table 6.4, where every applicable check passes and a cell reads n/a where a figure does not exercise a rule. The check earned its place during development, catching an unlabelled vertical axis on an early autocorrelation view before it reached this thesis.

Figure	**Title**	**Axes**	**Colour label**	**Uniform map**	**Overdraw**
Trace (FR-04)	pass	pass	n/a	n/a	pass
Autocorrelation	pass	pass	n/a	n/a	n/a
F-statistic (FR-09)	pass	pass	pass	pass	pass
PSD residual (FR-06)	pass	pass	n/a	n/a	pass
Waveform overlay	pass	pass	n/a	n/a	pass
Sky map (FR-02)	pass	n/a	pass	pass	pass
Waterfall	pass	pass	pass	pass	pass

Table 6.4: **Automatic-check results across the interactive figures.** The five mechanically verifiable heuristics of the computer assisted method (Zhu and Gumieniak 2021) run over the library's seven Plotly figures, grouped by module. Every check that applies to a figure passes. A cell reads n/a where the figure does not exercise that rule, such as a colour rule on a plot with no colour scale or the axis rule on the Sky Map whose axes are hidden by design. The corner plot renders through the static Matplotlib backend, which exposes no plot JSON, so it falls to the manual review below rather than to this table.

The same run also produced the method's advice output, the rules a program can detect but should not judge. On the Sky Map and the waterfall the function flagged that mark size carries a data variable, which is a reminder rather than a fault, because size has no intrinsic direction so the reader has to be told which way it points. The library uses size equal to one minus confidence, so a larger mark is a less certain source (Section 5.5). The advice rule exists to make sure that polarity is stated where the reader meets it rather than left to assumption. This is the method working as Zhu and Gumieniak intend, the program acting as a reminder of a known pitfall rather than a verdict on it.

The rules a program cannot check are left to manual heuristic review, which is the second evaluation method. A general usability checklist would not fit here. Standard heuristics such as Nielsen's widely used set were written for transactional interfaces, so they do not capture the concerns that dominate a data dense visualization: whether density reduction preserves the structure of the data, whether the colour and size encodings can be decoded and whether the mapping from physical parameter to visual mark is unambiguous. Dowding and Merrill (2018) make this same argument and build a heuristic set specific to dashboard visualizations, covering concerns such as data set reduction and information coding. That set provides the rubric for this manual review, so the interface was evaluated against it one heuristic at a time. Data set reduction is met by the Splatterplots-inspired density reduction on the Sky Map, which keeps the confusion foreground legible without discarding outlying resolvable sources. Information coding is met by the perceptually uniform colour maps, the encoding of frequency as colour and the encoding of source uncertainty as mark size. Clarity of mapping is met by keeping familiar idioms, so each

mark stands for a quantity the physicist already recognizes (Chapter 2 and Section 5.5). These are the judgements the automated checks deliberately leave alone, because they turn on whether a human reads the result correctly rather than on a property a program can read off the figure.

Beyond these two methods, Alhamadi et al. (2025) propose a behavioural indicator framework that derives usability signals from objective logs of user interaction such as click sequences and hover patterns. This gives a way to evaluate the interface without user self reporting. It would complement the heuristic review once the tool is deployed to enough users to collect such logs. A user study was considered and set aside, because the method has to match the artifact it evaluates. The contribution ranks the architecture first, the library second and the dashboard third, yet a user study exercises only the dashboard, which exists as a layout rather than a deployment, while the qualified user pool today is the handful of physicists who operate the pipeline, so a study drawn from it would return anecdote rather than evidence. Heuristic inspection is the established method for that situation (Paz and Pow-Sang 2016). The automated checks confirm that the figures satisfy concrete interface properties and the manual review judges whether the encodings can be read correctly, so what stays open is how physicists actually use the tool, a question for the behavioural method above once the dashboard is deployed.

6.4.1 Cognitive Load Reduction

The layout is designed to reduce the effort of cross referencing multiple output files. Drilling from a source on the macroscopic Sky Map to its sub-band and source slots, the drill down capability, removes the need to run separate `diagnostic.py` scripts for each source by hand (Basset 2024).

These choices answer failures the dashboard literature has already documented. Alhamadi et al. (2025) single out four recurring problems on visual analytics dashboards: information overload, inappropriate data order and grouping, ineffective data presentation and misaligned visual literacy. Each of these is something the user perceives on the screen, so the interface answers them through what it shows and how it arranges it rather than through the internal software structure that the analyst never sees. The Sky Map density abstraction and the drill down already described hold off information overload, because they keep the population overview legible and release per source detail only when the physicist asks for it. The core chart layout of Section 5.8 settles the question of data order and grouping, since it fixes the scan path rather than leaving the views to fall wherever they happen to land. The perceptually uniform encodings and the familiar idioms keep the presentation readable. Designing for an expert audience (Chapter 3) meets the literacy problem from the other side, since the danger here is rarely that a physicist cannot read a chart and more often that one of the tool's own rendering choices can be misread. The same study adds a caution worth keeping: the users who fall back on heavy filtering tend to be the ones already struggling, so a legible default view defends usability better than an abundance of filters. Confirming that the design avoids these failures in practice would mean recording how physicists actually move through the interface, which is the behavioural indicator method noted above as a planned extension rather than a measurement carried out here.

6.4.2 Trust and Transparency

Exposing the raw MCMC chains alongside the summary statistics, for example showing the trace plot next to the corner plot, is a deliberate transparency choice. It adheres to the "Trustworthy Design" principles (McKinley, Pandey, and Ottley 2025), allowing physicists to independently verify convergence rather than relying on the algorithm's output without scrutiny.

6.5 Summary

The evaluation shows that the visualization library addresses the gaps identified in Chapter 2. It meets its functional and non functional requirements, handles mission scale datasets without exhausting memory and gives the Global Fit pipeline an architectural foundation for moving from a collection of ad hoc scripts to a structured, interactive analysis environment. With LISA still more than a decade from launch, the worth of that foundation is in being early and solid enough for later work to extend as the pipeline matures and its data grows toward full mission scale.

Chapter 7

Conclusion

This dissertation worked on the software engineering and human computer interaction problems in the LISA Global Fit pipeline. The mission is preparing to process data streams with millions of overlapping gravitational wave signals, so it needs structured visualization tools rather than improvised ones (Basset 2024). The current reliance on ad hoc, tightly coupled plotting scripts is the practical bottleneck this work set out to remove, because those scripts make the outputs of trans-dimensional MCMC samplers hard for physicists to inspect and validate.

7.1 Summary of Contributions

This work followed a Design Science Research (DSR) methodology. Its central contribution is a reference architecture for visualizing the trans-dimensional Bayesian output of the LISA Global Fit, which is validated through two subsequent artifacts:

1. **A reference architecture:** The primary contribution separates data ingestion, a domain model and rendering into three decoupled layers. It hides the pipeline's heterogeneous storage formats (HDF5, NumPy and Pickle) behind a repository interface and represents the variable cardinality of trans-dimensional MCMC draws as first class domain objects. Expressed in language neutral patterns, it answers the second research question by proving how software engineering principles can map complex inference data into a maintainable visualization system.

2. **A validating library:** An object oriented Python library proves the architecture is buildable and meets non-functional constraints. It demonstrates that the decoupling holds in practice and that bounded streaming keeps peak memory within the limits of a standard workstation despite multi-gigabyte catalogs. By supplying density reduced sky maps, N dimensional corner plots and per walker traces, it answers the third research question regarding implementation strategies at mission scale.

3. **A layout demonstrating utility:** The library's modules are composed into a dashboard layout that applies the visualization and human factors knowledge surveyed in Chapter 2.

While currently a formal design, it confirms that the decoupled modules combine into the unified diagnostic interface the mission will eventually need.

The validating library is public. Its code, tests and runnable examples are available at https://github.com/PedroRomao3/lisaviz under the MIT license, so the artifact this thesis evaluates can be inspected, run and reused directly.

7.2 Limitations

The current implementation has limitations that should be acknowledged:

- **Memory Constraints for Full-Resolution Chains:** While the ingestion layer parses the central catalog efficiently, the full high resolution MCMC chains behind it are far larger, so pulling thousands of them into memory at once still strains a standard workstation. For now, the workaround is to either downsample the chains or to inspect the sources sequentially (loading and discarding one chain at a time) rather than holding the full population's high-resolution history in memory simultaneously.
- **Scope Limited to Galactic Binaries:** The current library implements the domain models and adapters exclusively for Galactic Binaries. While the Global Fit pipeline also resolves Massive Black Hole Binaries (MBHB) and fits the instrumental Noise model, these are not yet supported by the visualization tool. As established in Chapter 4, the architecture is explicitly designed to accept these as extensions, requiring only new adapters in the Data Ingestion Layer, but building out those components remains a necessary next step.
- **Evaluation Limits:** The requirements were verified by construction, measurement and inspection rather than through user testing, for the reasons given in Section 6.4. The memory bound was measured on the real Sangria catalogs and on synthetic catalogs served through the mock repository the architecture provides, while ingestion time and interactive frame rate have not been timed yet.

7.3 Future Work

Several directions follow on from this work:

- **Extension to All Source Types:** Directly addressing the scope limitation, future work must extend the library to support Massive Black Hole Binaries and the instrumental Noise model. Because the architecture was deliberately designed for this (Section 4.2), this extension does not require rewriting the core domain. It only requires adding new adapters to the Data Ingestion Layer and mapping their respective visual representations.
- **Implementing the Dashboard Layout:** The artifacts presented here run locally and the dashboard layout proposed in Section 5.8 is currently a formal design. A direct continuation

of this work is to implement that layout as a hosted web-based monitoring dashboard that the wider LISA consortium can reach without local setup. The decoupled, model driven foundation adopted here provides the exact separation of concerns needed to build such an interface on top of the existing domain logic without rewriting the scientific data processing layers.

- **Live Monitoring of Pipeline Progress:** The current library is designed for static, post-run analysis. A valuable future extension is to build a live monitoring view that updates dynamically while the Global Fit is running. To preserve the architecture's decoupling and avoid showing un-converged transient states, this live view cannot read the sampler's intermediate checkpoint files. Instead, it must operate at the iteration level. The Data Ingestion Layer could be extended to watch the versioned run directory (`gfrun/`) and automatically refresh the dashboard via the existing lazy loading adapters whenever the pipeline publishes a new, stable central catalog. Architecturally, this could be realized by implementing the **Observer** (or Publish-Subscribe) pattern, where the directory watcher acts as the publisher and the dashboard components act as subscribers that re-render when notified of a new iteration. This would let physicists safely watch the foreground resolve over time without breaking the decoupling boundary.

- **Lazy Loading and Out of Core Processing:** Directly addressing the memory constraints of full-resolution chains, the Data Ingestion Layer should move to out of core computation frameworks such as Dask or Vaex to lazily load and visualize data larger than local RAM as the mission approaches its terabyte scale datasets.

- **Machine Learning Diagnostics:** Future iterations could integrate anomaly detection algorithms to automatically classify failure modes in the MCMC chains (e.g., label switching vs. local minima trapping) before they are even visualized.

7.4 Final Remarks

The LISA mission is one of ESA's major scientific programs (ESA 2025), with launch planned for 2037. Its success will depend on the precision of the hardware in space and the mathematics of the inference algorithms, but it will also depend on the software that connects those algorithms to human understanding. This thesis takes an early and deliberate step toward that software: a structured visualization architecture that gives the Global Fit ground segment a foundation to build on, in place of the ad hoc scripts used today. Establishing that foundation now, a decade ahead of launch, is what lets the visualization effort grow with the mission on solid ground rather than be rebuilt later.

References

Abbott, B. P., et al. 2016. "Observation of Gravitational Waves from a Binary Black Hole Merger." *Phys. Rev. Lett.* 116 (6): 061102. https://doi.org/10.1103/PhysRevLett.116.061102. https://link.aps.org/doi/10.1103/PhysRevLett.116.061102.

Alhamadi, Mohammed. 2020. "Challenges, Strategies and Adaptations on Interactive Dashboards." In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization,* 368–371. UMAP '20. Genoa, Italy: Association for Computing Machinery. ISBN: 9781450368612. https://doi.org/10.1145/3340631.3398678. https://doi.org/10.1145/3340631.3398678.

Alhamadi, Mohammed, Hatim Alsayahani, Sarah Clinch, and Markel Vigo. 2025. "Behavioural Indicators of Usability in Visual Analytics Dashboards." *ACM Trans. Interact. Intell. Syst.* (New York, NY, USA) 15, no. 2 (April). ISSN: 2160-6455. https://doi.org/10.1145/3715710. https://doi.org/10.1145/3715710.

Amaro-Seoane, Pau, Heather Audley, Stanislav Babak, John Baker, Enrico Barausse, Peter Bender, Emanuele Berti, et al. 2017. *Laser Interferometer Space Antenna.* arXiv: 1702.00786 [astro-ph.IM]. https://arxiv.org/abs/1702.00786.

Anghelea, Anca, Manil Maskey, Shin-ichi Sobue, and Naoko Sugita. 2023. "THE NASA-ESA-JAXA EARTH OBSERVATION DASHBOARD." In *IGARSS 2023 - 2023 IEEE INTERNATIONAL GEOSCIENCE AND REMOTE SENSING SYMPOSIUM,* 418–420. IEEE International Symposium on Geoscience and Remote Sensing IGARSS. IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Pasadena, CA, JUL 16-21, 2023. IEEE; Inst Elect & Elect Engineers, Geoscience & Remote Sensing Soc. ISBN: 979-8-3503-2010-7. https://doi.org/10.1109/IGARSS52108.2023.10282443.

Bach, Benjamin, Euan Freeman, Alfie Abdul-Rahman, Cagatay Turkay, Saiful Khan, Yulei Fan, and Min Chen. 2023. "Dashboard Design Patterns." *IEEE Transactions on Visualization and Computer Graphics* 29 (1): 342–352. https://doi.org/10.1109/TVCG.2022.3209448.

Baskerville, Richard L. 1999. "Investigating Information Systems with Action Research." *Communications of the Association for Information Systems* 2 (1): 19. https://doi.org/10.17705/1CAIS.00219. https://aisel.aisnet.org/cais/vol2/iss1/19.

Basset, Antoine. 2024. *Processing LISA's data as a human: The GlobalFit Framework user experience.* https://pretalx.com/adass2024/talk/JNATUY/. Presented at Astronomical Data Analysis Software & Systems XXXIV (ADASS XXXIV), Aula Magna, Nov 2024.

Blanchette, Jasmin. 2008. *The Little Manual of API Design.* Technical report. Available at: https://people.mpi-inf.mpg.de/~jblanche/api-design.pdf. Trolltech/Nokia.

Bloch, Joshua. 2006. "How to Design a Good API and Why It Matters." In *Companion to the 21st ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '06),* 506–507. ACM. https://doi.org/10.1145/1176617.1176622.

Brehmer, Matthew, and Tamara Munzner. 2013. "A Multi-Level Typology of Abstract Visualization Tasks." *IEEE Transactions on Visualization and Computer Graphics* 19 (12): 2376–2385. https://doi.org/10.1109/TVCG.2013.124.

Brocke, Jan vom, Alan Hevner, and Alexander Maedche. 2020. "Introduction to Design Science Research," 1–13. September. ISBN: 978-3-030-46780-7. https://doi.org/10.1007/978-3-030-46781-4_1.

Carlin, Bradley P., and Siddhartha Chib. 1995. "Bayesian Model Choice Via Markov Chain Monte Carlo Methods." *Journal of the Royal Statistical Society: Series B (Methodological)* 57 (3): 473–484. ISSN: 0035-9246. https://doi.org/10.1111/j.2517-6161.1995.tb02042.x. https://doi.org/10.1111/j.2517-6161.1995.tb02042.x.

Cécile Cavet. 2022. *LISA Data Challenge 2a: Sangria Dataset.* https://lisa-ldc.in2p3.fr/challenge2a. Accessed: 2026-01-05.

Collette, A. 2013. *Python and HDF5: Unlocking Scientific Data.* O'Reilly Media. ISBN: 9781491945018. https://books.google.pt/books?id=lRCMAQAAQBAJ.

Crowder, Jeff, and Neil J. Cornish. 2007. "A solution to the galactic foreground problem for LISA." *Physical Review D* 75 (4): 043008. https://doi.org/10.1103/PhysRevD.75.043008. https://link.aps.org/doi/10.1103/PhysRevD.75.043008.

Deng, Senwen, Stanislav Babak, Maude Le Jeune, Sylvain Marsat, Éric Plagnol, and Andrea Sartirana. 2025. "Modular global-fit pipeline for LISA data analysis." *Phys. Rev. D* 111 (May): 103014. ISSN: 2470-0010. https://doi.org/10.1103/PhysRevD.111.103014.

Dijkstra, Edsger W. 1982. "On the Role of Scientific Thought." In *Selected Writings on Computing: A Personal Perspective,* 60–66. Original manuscript EWD447, 1974. Springer.

Dowding, Dawn, and Jacqueline A. Merrill. 2018. "The Development of Heuristics for Evaluation of Dashboard Visualizations" [in en]. *Applied Clinical Informatics* 09, no. 03 (July): 511–518. ISSN: 1869-0327. https://doi.org/10.1055/s-0038-1666842. http://www.thieme-connect.de/DOI/DOI?10.1055/s-0038-1666842.

ESA. 2025. *LISA – Laser Interferometer Space Antenna.* https://www.cosmos.esa.int/web/lisa.

European Cooperation for Space Standardization. 2025. *Active Standards.* https://ecss.nl/standards/active-standards/.

Evans, Eric. 2003. *Domain-Driven Design: Tackling Complexity in the Heart of Software.* Addison-Wesley. ISBN: 978-0321125217.

Few, Stephen. 2006. *Information dashboard design: the effective visual communication of data* [in eng]. 1. Aufl. Sebastopol, CA: O'Reilly & Associates. ISBN: 9780596100162.

Foreman-Mackey, Daniel. 2016. "corner.py: Scatterplot matrices in Python." *Journal of Open Source Software* 1 (2): 24. https://doi.org/10.21105/joss.00024.

Foreman-Mackey, Daniel, David W. Hogg, Dustin Lang, and Jonathan Goodman. 2013. "emcee: The MCMC Hammer." *Publications of the Astronomical Society of the Pacific* 125 (925): 306–312. https://doi.org/10.1086/670067.

Fowler, Martin. 2002. *Patterns of Enterprise Application Architecture.* Addison-Wesley. ISBN: 978-0321127426.

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley. ISBN: 978-0201633610.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2013. *Bayesian Data Analysis.* 3rd ed. Chapman / Hall/CRC. ISBN: 9781439840955.

Gomes, Tiago, Carlos M. Correia, Lisa Bardou, Sylvain Cetre, Johann Kolb, Caroline Kulcsár, François Leroux, et al. 2024. "Adaptive optics telemetry standard: Design and specification of a novel data exchange format." *Astronomy & Astrophysics* 686:A7. https://doi.org/10.1051/0004-6361/202348486.

Goodman, Jonathan, and Jonathan Weare. 2010. "Ensemble Samplers with Affine Invariance." *Communications in Applied Mathematics and Computational Science* 5 (1): 65–80. https://doi.org/10.2140/camcos.2010.5.65.

Green, Peter J. 1995. "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination." *Biometrika* 82 (4): 711–732. https://doi.org/10.1093/biomet/82.4.711. https://doi.org/10.1093/biomet/82.4.711.

Hastings, W. K. 1970. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications." *Biometrika* 57 (1): 97–109. https://doi.org/10.1093/biomet/57.1.97.

Hevner, Alan R., Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. "Design Science in Information Systems Research." *Management Information Systems Quarterly* 28 (March): 75–105.

Iivari, Juhani, and John R. Venable. 2009. "Action research and design science research - Seemingly similar but decisively dissimilar." In *ECIS 2009 Proceedings,* 73. European Conference on Information Systems (ECIS). http://aisel.aisnet.org/ecis2009/73.

Islam, Mohaiminul, and Shangzhu Jin. 2019. "An Overview of Data Visualization." In *2019 International Conference on Information Science and Communications Technologies (ICISCT),* 1–7. November. https://doi.org/10.1109/ICISCT47635.2019.9012031. https://ieeexplore.ieee.org/document/9012031.

ISO/IEC. 2016. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality.* 25023:2016. ISO/IEC, June. https://www.iso.org/obp/ui/#iso:std:iso-iec:25023:ed-1:v1:en.

———. 2023. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model.* 25010:2023. ISO/IEC, November. https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-2:v1:en.

Jr., D. C. Fargas, R. M. de la Cruz, and A. C. Blanco. 2024. "SPACE+ DATA DASHBOARD: EMPOWERING INSTITUTIONS AND CITIZENS WITH SEAMLESS SPACE DATA ACCESS." *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* https://api.semanticscholar.org/CorpusID:269417195.

Karnesis, Nikolaos, Michael L. Katz, Natalia Korsakova, Jonathan R. Gair, and Nikolaos Stergioulas. 2023. "Eryn: a multipurpose sampler for Bayesian inference." *Monthly Notices of the Royal Astronomical Society* 526, no. 4 (September): 4814–4830. ISSN: 1365-2966. https://doi.org/10.1093/mnras/stad2939.

Lin, Yanna, Haotian Li, Aoyu Wu, Yong Wang, and Huamin Qu. 2024. "DMiner: Dashboard Design Mining and Recommendation." *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS* 30, no. 7 (July): 4108–4121. ISSN: 1077-2626. https://doi.org/10.1109/TVCG.2023.3251344.

LISA DDPC. 2026a. *Description of the Common Dataset 1 Light (Mojito Light) of the LISA DDPC.* Technical Note LISA-DDPC-SEG-TN-008. Revision 00 02. LISA Distributed Data Processing Center, May.

———. 2026b. *L2 to L3 model.* Technical Note. Date: 8 June 2026. LISA Distributed Data Processing Center, June.

Martin, Osvaldo A., Oriol Abril-Pla, Jordan Deklerk, Seth D. Axen, Colin Carroll, Ari Hartikainen, and Aki Vehtari. 2026. "ArviZ: a modular and flexible library for exploratory analysis of Bayesian models." *Journal of Open Source Software* 11 (119): 9889. https://doi.org/10.21105/joss.09889. https://doi.org/10.21105/joss.09889.

Martin, Robert C. 2017. *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* Prentice Hall. ISBN: 978-0134494166.

Mayorga, Adrian, and Michael Gleicher. 2013. "Splatterplots: Overcoming Overdraw in Scatter Plots." *IEEE transactions on visualization and computer graphics* 19, no. 9 (September): 1526–1538. ISSN: 1077-2626. https://doi.org/10.1109/TVCG.2013.65. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4048834/.

McKinley, Oen G, Saugat Pandey, and Alvitta Ottley. 2025. "Trustworthy by Design: The Viewer's Perspective on Trust in Data Visualization." In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems.* CHI '25. Association for Computing Machinery. ISBN: 9798400713941. https://doi.org/10.1145/3706598.3713824. https://doi.org/10.1145/3706598.3713824.

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. "Equation of State Calculations by Fast Computing Machines." *The Journal of Chemical Physics* 21 (6): 1087–1092. https://doi.org/10.1063/1.1699114.

Midway, Stephen R. 2020. "Principles of Effective Data Visualization." *Patterns* 1 (9): 100141. ISSN: 2666-3899. https://doi.org/10.1016/j.patter.2020.100141. https://www.sciencedirect.com/science/article/pii/S2666389920301896.

Miller, Robert B. 1968. "Response time in man-computer conversational transactions." In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I,* 267–277. AFIPS '68 (Fall, part I). San Francisco, California: Association for Computing Machinery. ISBN: 9781450378994. https://doi.org/10.1145/1476589.1476628. https://doi.org/10.1145/1476589.1476628.

NASA. 2016. *NASA Systems Engineering Handbook.* https://www.nasa.gov/reference/systems-engineering-handbook/. NASA/SP-2016-6105 Rev2.

Nunamaker, J.F., and M. Chen. 1990. "Systems development in information systems research." In *Twenty-Third Annual Hawaii International Conference on System Sciences,* vol. 3, 631–640 vol.3. https://doi.org/10.1109/HICSS.1990.205401.

Parnas, David L. 1972. "On the Criteria to Be Used in Decomposing Systems into Modules." *Communications of the ACM* 15, no. 12 (December): 1053–1058. https://doi.org/10.1145/361598.361623.

Parsons, Paul. 2022. "Understanding Data Visualization Design Practice." *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS* 28, no. 1 (January): 665–675. ISSN: 1077-2626. https://doi.org/10.1109/TVCG.2021.3114959.

Paz, Freddy, and José Antonio Pow-Sang. 2016. "A Systematic Mapping Review of Usability Evaluation Methods for Software Development Process." *International Journal of Software Engineering and Its Applications* 10, no. 1 (January): 165–178. ISSN: 17389984, 17389984. https://doi.org/10.14257/ijseia.2016.10.1.16. http://www.sersc.org/journals/IJSEIA/vol10_no1_2016/16.pdf.

Peffers, Ken, Tuure Tuunanen, Marcus Rothenberger, and S. Chatterjee. 2007. "A design science research methodology for information systems research." *Journal of Management Information Systems* 24 (January): 45–77.

Pham, Timothy T., and Jason Liao. 2008. "DSN Performance Dashboards," https://api.semanticscholar.org/CorpusID:108525250.

Ramachandran, Prabhu, and Gael Varoquaux. 2011. "Mayavi: 3D Visualization of Scientific Data." *Computing in Science & Engineering* 13 (2): 40–51. https://doi.org/10.1109/MCSE.2011.35.

Robitaille, Thomas P., Erik J. Tollerud, Perry Greenfield, Michael Droettboom, Erik Bray, Tom Aldcroft, Matt Davis, et al. n.d. "Astropy: A community Python package for astronomy." *Astronomy & Astrophysics* 558 (September): A33. ISSN: 1432-0746. https://doi.org/10.1051/0004-6361/201322068. http://dx.doi.org/10.1051/0004-6361/201322068.

Sarikaya, Alper, Michael Correll, Lyn Bartram, Melanie Tory, and Danyel Fisher. 2019. "What Do We Talk About When We Talk About Dashboards?" *IEEE Transactions on Visualization and Computer Graphics* 25, no. 1 (January): 682–692. ISSN: 1077-2626. https://doi.org/10.1109/TVCG.2018.2864903.

Sedrakyan, Gayane, Erik Mannens, and Katrien Verbert. 2019. "Guiding the choice of learning dashboard visualizations: Linking dashboard design and data visualization concepts." *Journal of Computer Languages* 50 (February): 19–38. ISSN: 2590-1184. https://doi.org/10.1016/j.jvlc.2018.11.002. https://www.sciencedirect.com/science/article/pii/S1045926X18301009.

Setlur, Vidya, Michael Correll, Arvind Satyanarayan, and Melanie Tory. 2024. "Heuristics for Supporting Cooperative Dashboard Design." *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS* 30, no. 1 (January): 370–380. ISSN: 1077-2626. https://doi.org/10.1109/TVCG.2023.3327158.

Shi, Ting, Pengyu Li, Wu Yang, Ailin Qi, and Junfei Qiao. 2023. "Application of TCN-biGRU neural network in $PM_{2.5}$ concentration prediction." *Environmental Science and Pollution Research* 30:1–12. https://doi.org/10.1007/s11356-023-30354-6.

Snyder, John P. 1987. *Map Projections: A Working Manual.* U.S. Geological Survey Professional Paper 1395. Washington, D.C.: United States Government Printing Office. https://doi.org/10.3133/pp1395.

Stevens, Wayne P., Glenford J. Myers, and Larry L. Constantine. 1974. "Structured Design." *IBM Systems Journal* 13 (2): 115–139. https://doi.org/10.1147/sj.132.0115.

Szafir, Danielle Albers. 2018. "Modeling Color Difference for Visualization Design." *IEEE Transactions on Visualization and Computer Graphics* 24, no. 1 (January): 392–401. ISSN: 1941-0506. https://doi.org/10.1109/TVCG.2017.2744359. https://ieeexplore.ieee.org/document/8017604.

Toasa, Renato M., Marisa da Silva Maximiano, Catarina I. Reis, and David Guevara. 2018. "Data visualization techniques for real-time information — A custom and dynamic dashboard for analyzing surveys' results." *2018 13th Iberian Conference on Information Systems and Technologies (CISTI),* 1–7. https://api.semanticscholar.org/CorpusID:49538490.

Vuckovic, Milena, Dawid Wolosiuk, and Johanna Schmidt. 2024. "Designing Interactive Analytics Dashboards for Diverse Target Groups: The Process and Decision-Making." In *2024 9th International Conference on Smart and Sustainable Technologies (SpliTech),* 1–4. June. https://doi.org/10.23919/SpliTech61897.2024.10612622. https://ieeexplore.ieee.org/document/10612622.

Walny, Jagoda, Christian Frisson, Mieka West, Doris Kosminsky, Søren Knudsen, Sheelagh Carpendale, and Wesley Willett. 2020. "Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff." *IEEE Transactions on Visualization and Computer Graphics* 26 (1): 12–22. https://doi.org/10.1109/TVCG.2019.2934538.

Wang, Zezhong, Lovisa Sundin, Dave Murray-Rust, and Benjamin Bach. 2020. "Cheat Sheets or Data Visualization Techniques." In *PROCEEDINGS OF THE 2020 CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS (CHI'20).* CHI Conference on Human Factors in Computing Systems (CHI), ELECTR NETWORK, APR 25-30, 2020. Assoc Comp Machinery; ACM SIGCHI. ISBN: 978-1-4503-6708-0. https://doi.org/10.1145/3313831.3376271.

Wilson, Greg, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K. Teal. 2016. "Good Enough Practices in Scientific Computing." *CoRR* abs/1609.00037. arXiv: 1609.00037. http://arxiv.org/abs/1609.00037.

Yi, Ji Soo, Youn ah Kang, John Stasko, and J.A. Jacko. 2007. "Toward a Deeper Understanding of the Role of Interaction in Information Visualization." *IEEE Transactions on Visualization and Computer Graphics* 13, no. 6 (November): 1224–1231. ISSN: 1941-0506. https://doi.org/10.1109/TVCG.2007.70515. https://ieeexplore.ieee.org/document/4376144.

Yin, Robert K. 2018. *Case Study Research and Applications: Design and Methods.* 6th. Thousand Oaks, CA: SAGE Publications.

Zhang, Nuowen, Jing Zhang, Shangsong Jiang, and Weijia Ge. 2024. "The Effects of Layout Order on Interface Complexity: An Eye-Tracking Study for Dashboard Design." *Sensors* 24 (18). ISSN: 1424-8220. https://doi.org/10.3390/s24185966. https://www.mdpi.com/1424-8220/24/18/5966.

Zhu, Ying, and Julia A. Gumieniak. 2021. "Computer-Assisted Heuristic Evaluation of Data Visualization." In *ADVANCES IN VISUAL COMPUTING (ISVC 2021), PT II,* edited by G Bebis, V Athitsos, T Yan, M Lau, F Li, C Shi, X Yuan, C Mousas, and G Bruder, 13018:408–420. Lecture Notes in Computer Science. 16th International Symposium on Visual Computing (ISVC), ELECTR NETWORK, OCT 04-06, 2021. ISBN: 978-3-030-90436-4; 978-3-030-90435-7. https://doi.org/10.1007/978-3-030-90436-4_33.

Appendix A

Candidate Profile

This annex lists the author's background to ground the software engineering skills applied in this thesis.

A.1 Academic Background

The author is completing a Master's degree in Informatics and Computing Engineering at the Faculty of Engineering of the University of Porto (FEUP). During the dissertation period and also previously, the author served as a Teaching Assistant at FEUP.

The author also completed an Erasmus+ exchange program at Politecnico di Torino (POLITO) in Turin, Italy.

A.2 Experience

The author's technical background centers on embedded systems and software architecture. Past experience includes:

- **Formula Student FEUP:** Participated in the team throughout the degree with increasing responsibility, serving eventually as Lead of the Embedded Software Sub-system to architect the vehicle's software for international competitions.
- **INESC TEC:** Worked as a Research Intern developing state estimation algorithms for autonomous vehicles.

A.3 Future Roles

Following this thesis, the author will join **Critical Flytech**, a joint venture between Airbus and Critical Software, as an Embedded Software Engineer. The role involves developing critical embedded software for the aerospace sector, relying on strict engineering standards similar to those demonstrated here.