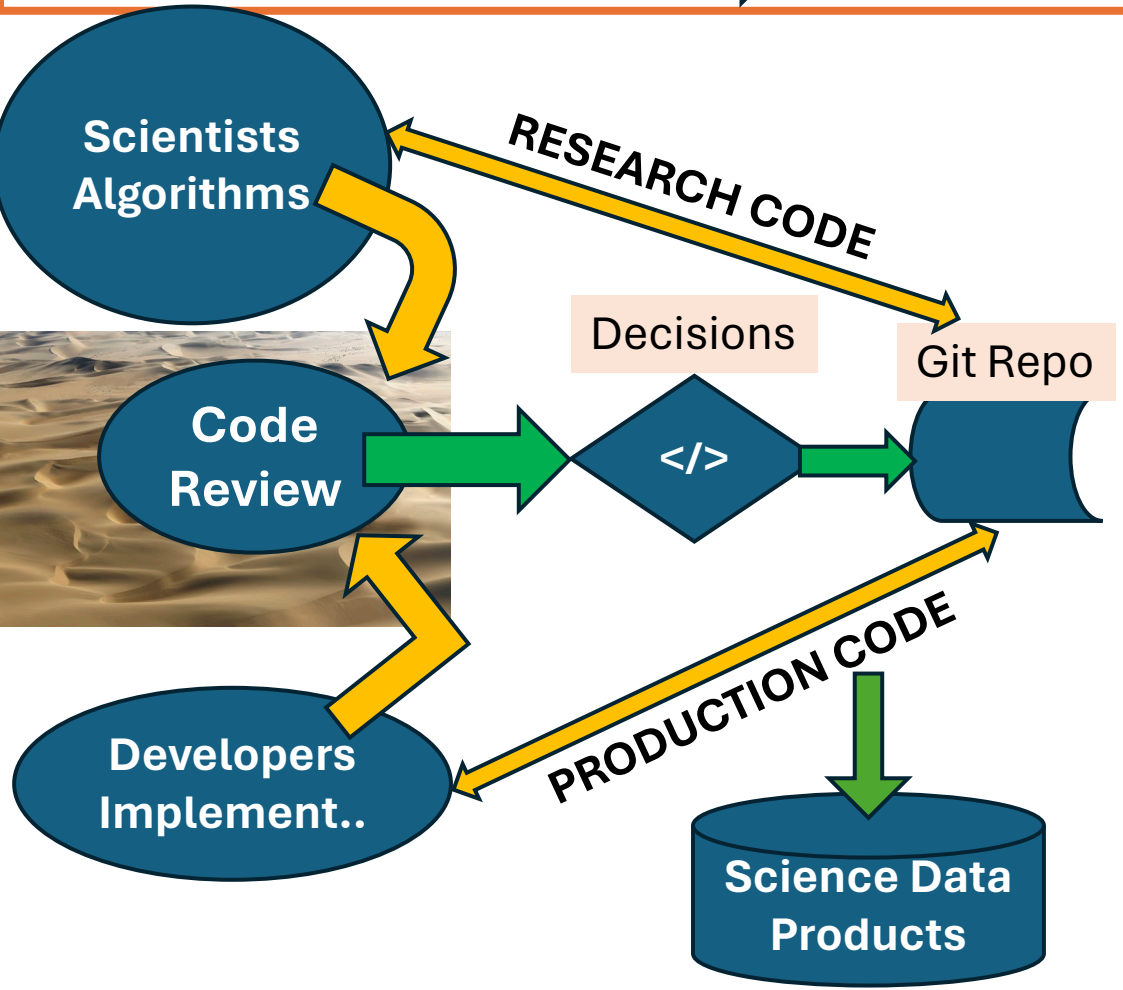
Fostering better collaboration in software development cycles between scientists and programmers to ensure the integrity of and promote the development of new scientific data products

- A. Gopalan[1,2], D.R. Doelling[2]

- 1- Analytical Mechanics Associates, Inc., Hampton, Virginia
- 2-NASA Langley Research Center, Hampton, Virginia
- Contact email: arun.gopalan-1@nasa.gov

- Abstract 2B-1

CERES/TISA Product Team etc. → (Climate Quality Data Record of Observed TOA Fluxes)



Version Control
Code Documentation
Testing/Validation
Continuous Int.
Reproducibility
Hi-Performance Comp
Interoperability & Data
Management
Software Sustainability

Research code and Production code serve different purposes and are developed with different goals and constraints

Research Code	Production Code
Developed to explore ideas, test hypotheses, and conduct experiments in scientific research, or prototyping. It is used in research projects, academic papers, and experimental studies where flexibility, experimentation, and rapid iteration are prioritized over stability and performance.	Developed for deployment in operational environments to provide reliable science data products. It is used where stability, performance, and scalability are critical.
May have loosely defined or evolving requirements, as it is often exploratory and experimental in nature. It may prioritize flexibility, ease of experimentation, and rapid prototyping over strict adherence to software engineering principles.	Typically has well-defined requirements, specifications, and quality standards. It must meet functional and non-functional requirements, such as reliability, security, performance, scalability, and maintainability.

Research Code	Production Code
Code may be less stable and reliable, as it is often developed iteratively to explore new ideas or approaches. It may contain experimental features, incomplete implementations, or temporary workarounds to test hypotheses or conduct experiments	Code is expected to be stable, reliable, and maintainable over time. It undergoes rigorous testing, validation, and quality assurance processes to ensure correctness, robustness, and resilience to failures.
Code may have less emphasis on documentation and code quality, as it is often developed by individual researchers or small teams with limited resources and time constraints. However, good documentation and code organization can still improve collaboration, reproducibility, and usability.	Code typically requires comprehensive documentation, clean code practices, and adherence to coding standards and best practices. It often undergoes code reviews and documentation reviews to ensure clarity, readability, and maintainability.

Research Code	Production Code
Code may have a shorter lifecycle, as it is often developed for specific research projects or experiments. It may be archived or deprecated once the research objectives are achieved, although some research code may be extended, reused, or adapted for future studies.	Code is intended for long-term use and maintenance. It may have a longer lifecycle, with regular updates, bug fixes, and support services to address user needs and evolving requirement

Modern Software Development Framework

- **Version Control**: Effective use of version control systems like Git to track changes, collaborate with other researchers, and maintain a history of code modifications.
- **Code Documentation**: Thorough documentation of code using tools like Doxygen, Sphinx, or Javadoc. This includes inline comments, function/method documentation, and overall project documentation to enhance code readability and usability.
- **Testing:** Adoption of automated testing practices such as unit testing, integration testing, and regression testing to ensure software correctness, reliability, and robustness. Test-driven development (TDD) or behavior-driven development (BDD) may be used to drive software design through test cases.
- **Continuous Integration/Continuous Deployment (CI/CD):** Integration of CI/CD pipelines to automate build, test, and deployment processes, enabling rapid iteration and delivery of software updates while maintaining quality and stability. (Agile)

Modern Software Development Framework

- **High-Performance Computing (HPC):** Optimization of algorithms and parallelization techniques to leverage the computational power of modern HPC systems, including multi-core CPUs, GPUs, and distributed computing clusters.
- **Reproducibility and Replicability**: Implementation of practices to ensure reproducibility and replicability of scientific results, including code and data sharing, containerization (e.g., Docker), and workflow management systems (e.g., Nextflow, Snakemake).
- **Interoperability and Data Management:** Adoption of open standards and formats for data exchange and interoperability. Integration with existing tools, libraries, and data repositories to facilitate collaboration and reuse.
- **Software Sustainability**: Consideration of long-term software sustainability and maintenance. This includes fostering a culture of open-source development, community engagement, and funding support for ongoing maintenance and development efforts.

Modern Software Development Framework

- **User Experience (UX):** Designing software with a focus on user experience to ensure usability, accessibility, and user satisfaction. This may involve user-centered design principles, user feedback mechanisms, and intuitive graphical interfaces.
- **Security and Privacy:** Implementation of security measures to protect sensitive data and prevent unauthorized access or tampering. This includes secure coding practices, encryption, access control, and compliance with relevant regulations and standards.
- Overall, modern scientific software development emphasizes collaboration, transparency, reproducibility, and usability to advance scientific research and innovation in various domains.

Specific CERES/TISA Issues

- Use of GIT repositories
- e.g. Narrow Band Radiance to Broad Band Flux Calculations
- Standardized LUT formats and read packages/ Calibration Provenance
- Common GEO Data Extraction/Gridding/Variable Spatial/Temporal Resolutions
- Ease of Debugging
- End-to-End Unit Testing
- Scaling of input sizes/Program Run times/ Parallelization where appropriate.
- Scientist assumes perfect valid inputs to Algorithm but in practice …
- No ad-hoc single use program modules
- Future Proofing to some extent