When Scientific Software Meets Software Engineering

Software Engineering for Scientific Computing

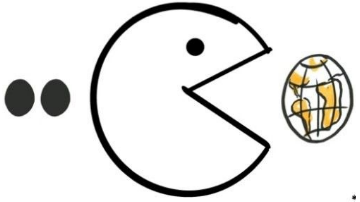
Prof. Benoit Combemale
University of Rennes 1
DiverSE team (IRISA & Inria)

http://combemale.fr / @bcombemale

“Software Is Eating the World”

Digitalization of our society

- personal context (health, music, video, social networks…)
- professional context (digitalization of numerous processes and activities)



THE WALL STREET JOURNAL.
WSJ.com
ESSAY | AUGUST 20, 2011

Why Software Is Eating The World

By MARC ANDREESSEN

This week, Hewlett-Packard (where I am on the board) announced that it is exploring jettisoning its struggling PC business in favor of investing more heavily in software, where it sees better potential for growth. Meanwhile, Google plans to buy up the cellphone handset maker Motorola Mobility. Both moves surprised the tech world. But both moves are also in line with a trend I've observed, one that makes me optimistic about the future growth of the American and world economies, despite the recent turmoil in the stock market.

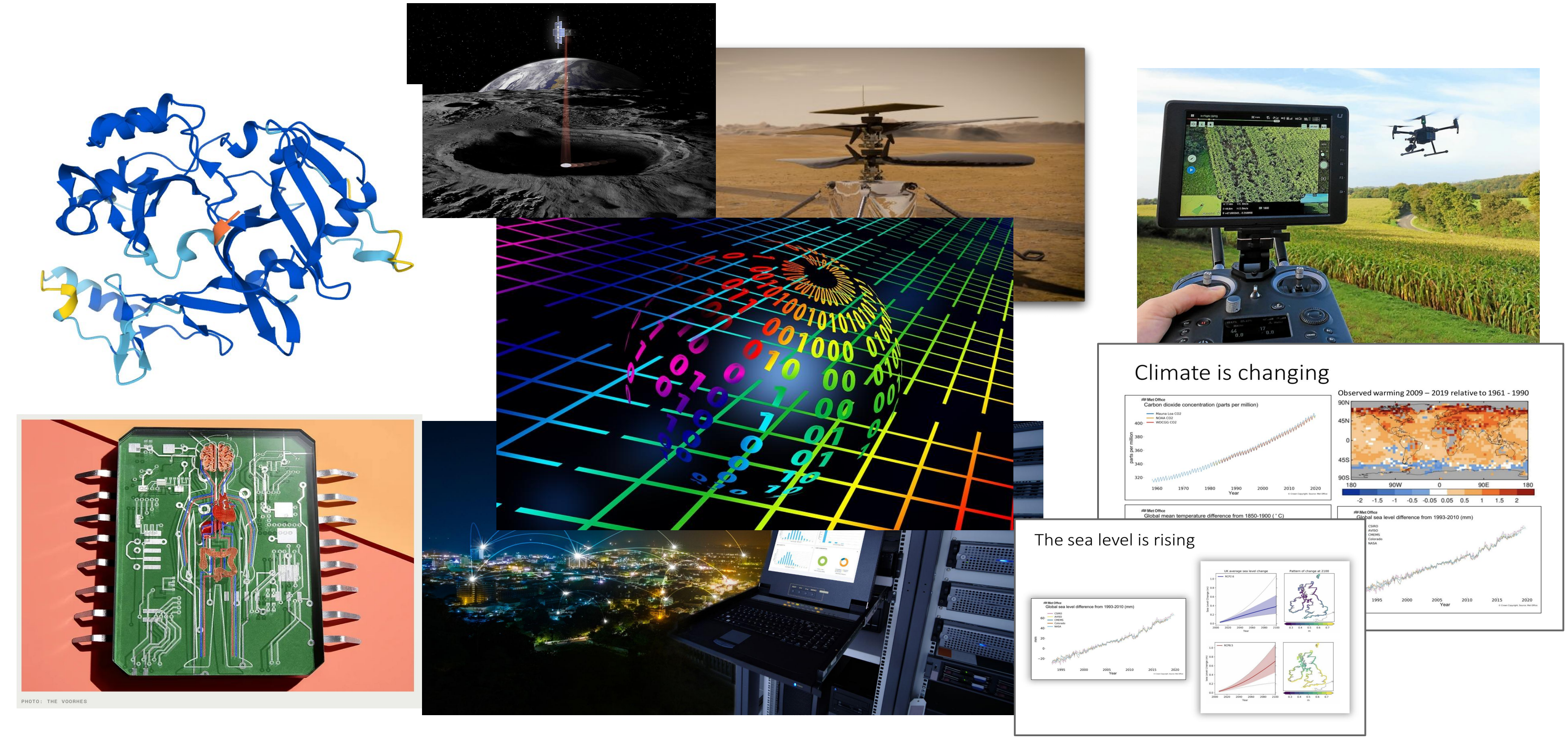
n interview with WSJ's Kevin Delaney, Groupon and edIn investor Marc Andreessen insists that the ent popularity of tech companies does not constitute bble. He also stressed that both Apple and Google undervalued and that "the market doesn't like tech."

In short, software is eating the world.

More than 10 years after the peak of the 1990s dot-com bubble, a dozen or so new Internet companies like Facebook and Twitter are sparking controversy in Silicon Valley, due to their rapidly growing private market valuations, and even the occasional successful IPO. With scars from the heyday of Webvan and Pets.com still fresh in the investor psyche, people are asking, "Isn't this just a dangerous new bubble?"

I, along with others, have been arguing the other side of the case. (I am co-founder and general partner of venture capital firm Andreessen-Horowitz, which has invested in Facebook, upon, Skype, Twitter, Zynga, and Foursquare, among others. I am also personally an investor in LinkedIn.) believe that many of the prominent new Internet companies are building real, high-growth, high-margin, ly defensible businesses.

(Scientific) Software Is Eating the World



Scientific Computing

In **science**

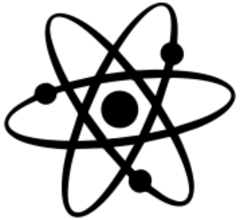
- holistic view of wicked problems
- what-if scenarios

In **industry**

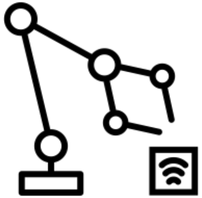
- design of complex systems
- digital twins

In **society**

- decision and policy makers
- broader engagement
- education



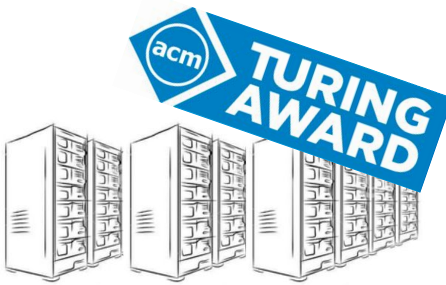
Sound knowledge



Innovative system

Smart people

Scientific Computing



- Use of **advanced computing capabilities**

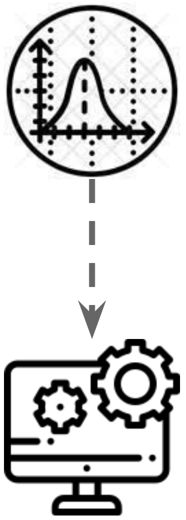
- To **understand** and **solve**
  - scientific problems (e.g., biological, physical, and social),
  - engineering problems, and
  - humanities problems.

- And **predict** the behavior or the outcome of a physical system, being natural or man-made.

Scientific Software

*Rely on the development of **mathematical models** to understand physical systems through their **simulations**.*

- Mathematical models belong to numerical models (continuous or discrete) and analytics.
- Simulations of mathematical models correspond to the execution of the computer programs containing these models, the so-called ``simulation codes".
- Scientific software = software dedicated to scientific computing and simulation.

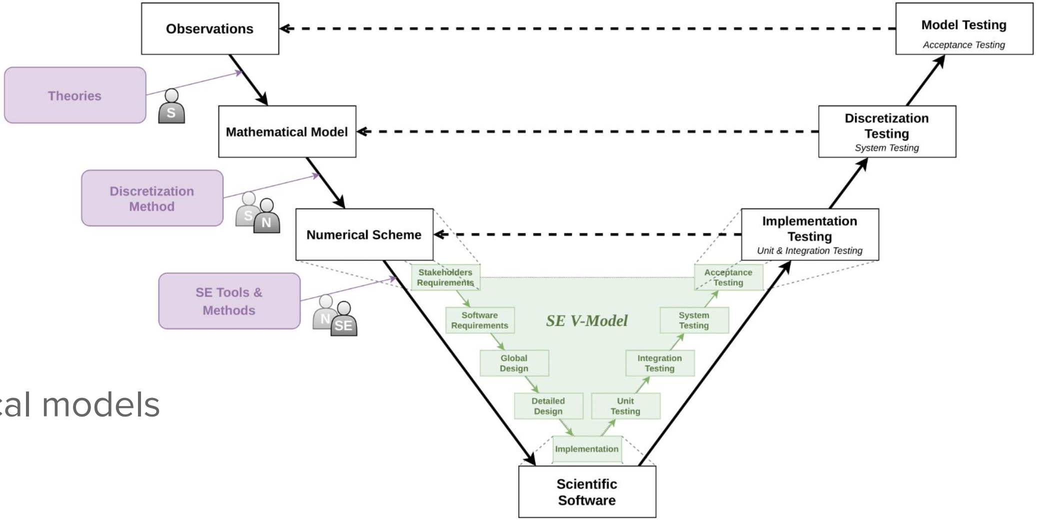


Scientific Software Specificities

- Specific **software**
  - highly iterative process,
  - large dataset,
  - complex compilation chains and deployments,
  - long lifetime application,
  - *usually, a means not an end.*
- Specific **stakeholders**
  - developers: scientists (physicist, mathematicians…), engineers, numerical analyst…
  - end-users: from the developper itself to decision makers and the general public
  - funders: possibly third parties, without any background on software development
  - …

Scientific Software Development

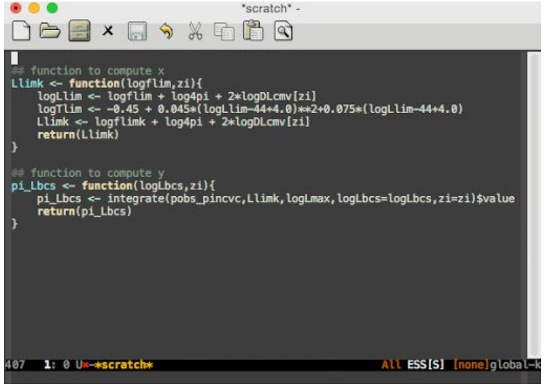
- Scientific V-Model subsumes SE V-Model
- Multidisciplinary development
- Collaborative (time/space) development
- Highly configurable / variable mathematical models



When Scientific Software Meets Software Engineering
D. Leroy, J. Sallou, J. Bourcier and B. Combemale. Computer, vol. 54, no. 12, pp. 60-71, 2021.
Preprint: https://hal.inria.fr/hal-03318348v1

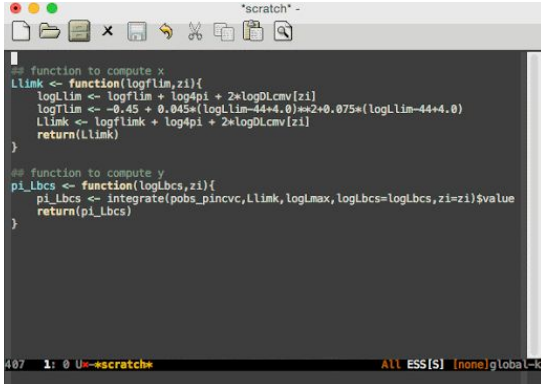
From Scientific Coding…

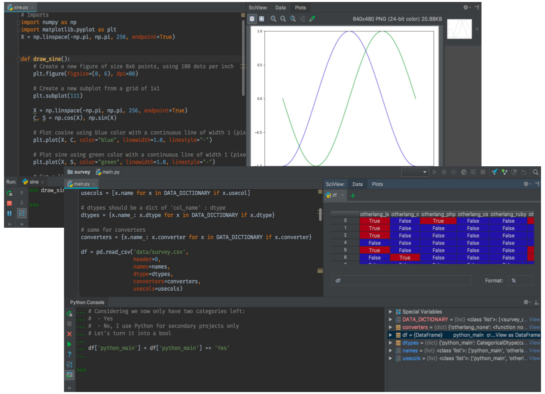
~Syntactic support



From Scientific Coding…

~Syntactic support



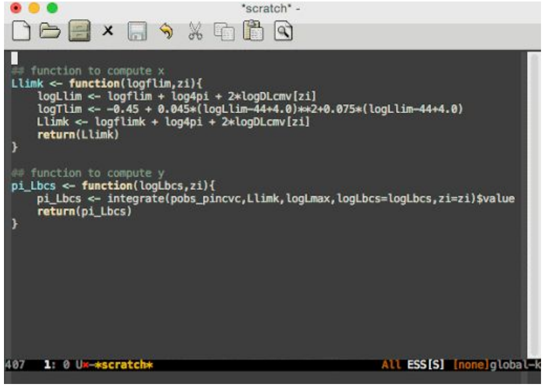


… to Structured and Sound Programming

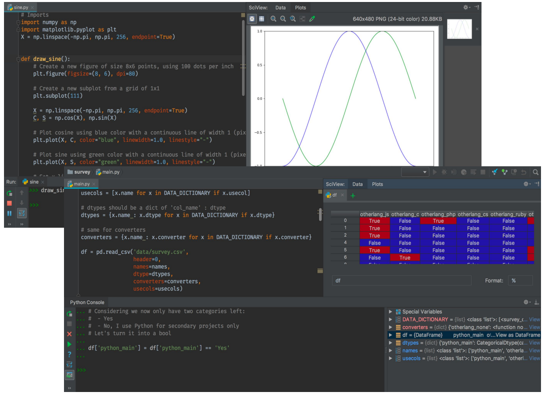
- **Abstractions** (modularity, resources, computation…)
- **Automation** (dev/doc/test, compilation/integration, deployment, delivery…)
- **Validation & Verification**

From Scientific Coding…

~Syntactic support



Are they really fitting specificities of scientific software?

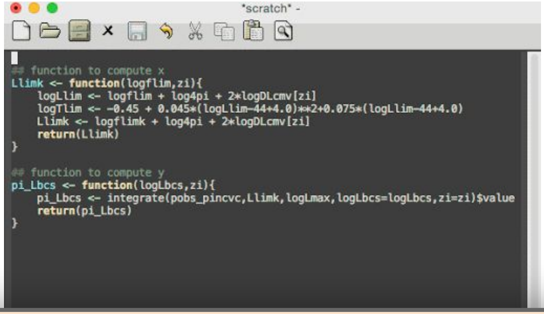


… to Structured and Sound Programming

- **Abstractions** (modularity, resources, computation…)
- **Automation** (dev/doc/test, compilation/integration, deployment, delivery…)
- **Validation & Verification**

From Scientific Coding...

~Syntactic support

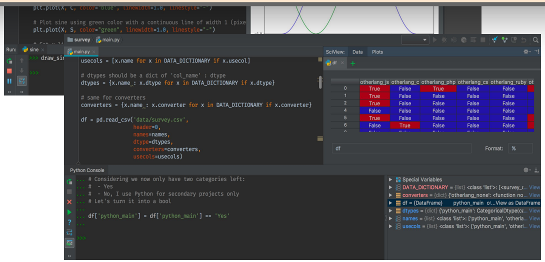


Are they really fitting specificities of ... ware?

Software engineering tools & methods are mostly develop by/for software engineers:

- require a strong background in software engineering
- target software as a final product

...l and ...ming



- **Abstractions** (modularity, resources, computation...)
- **Automation** (dev/doc/test, compilation/integration, deployment, delivery...)
- **Validation & Verification**

Momentum

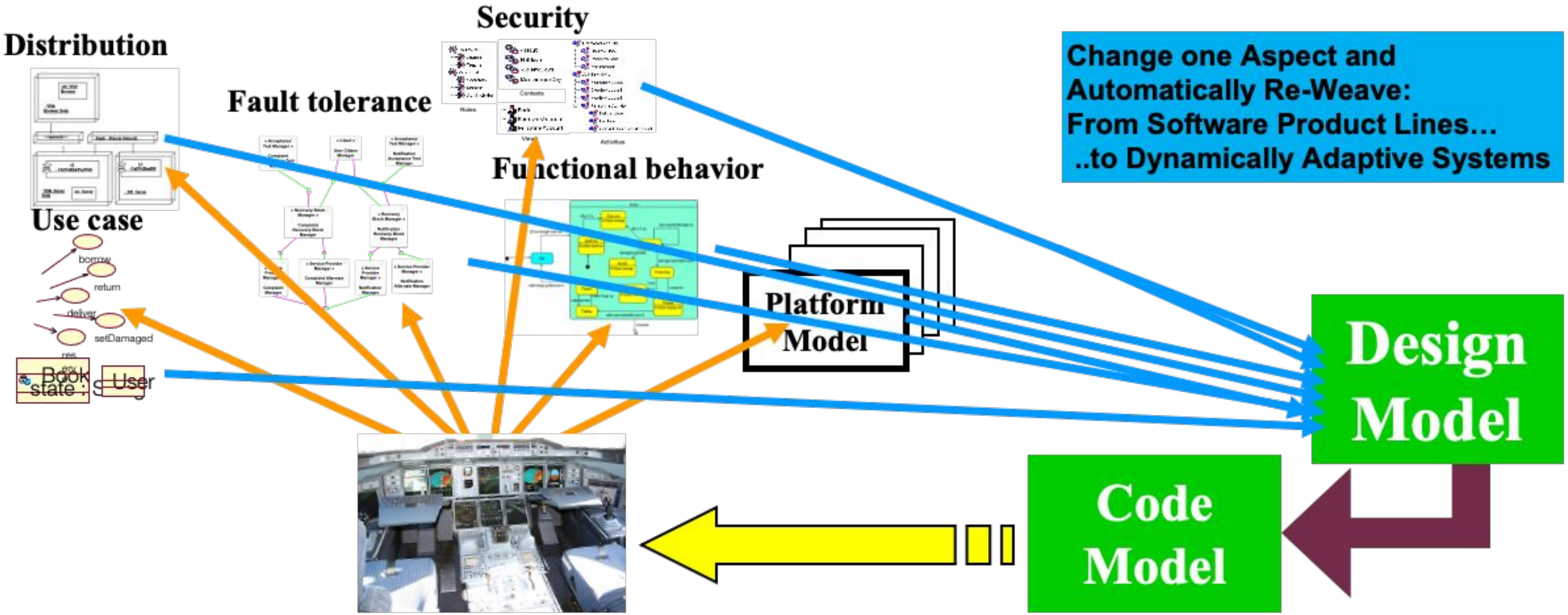
- Ever-increasing intrinsic complexity of mathematical models
- Broader engagement of various heterogeneous stakeholders
- Numerous scenarios to evaluate
- Ever more efficient and large simulations

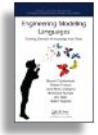
⇒ It urges to establish the required software engineering foundations, tools and methods for scientific computing

⇒ The SE4Science Initiative!

On Software Languages

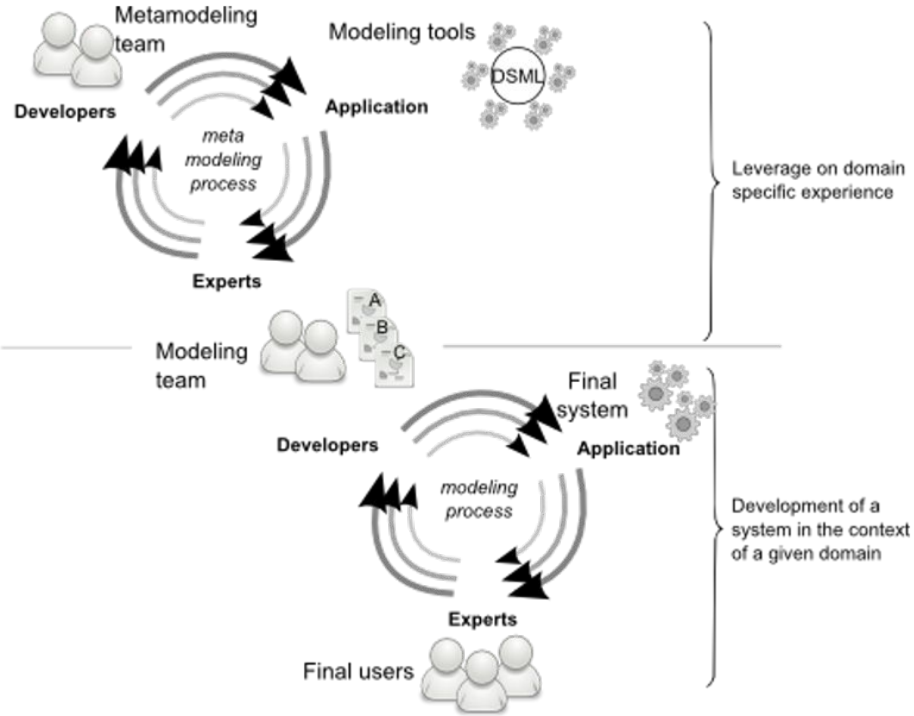
Model-Driven Engineering





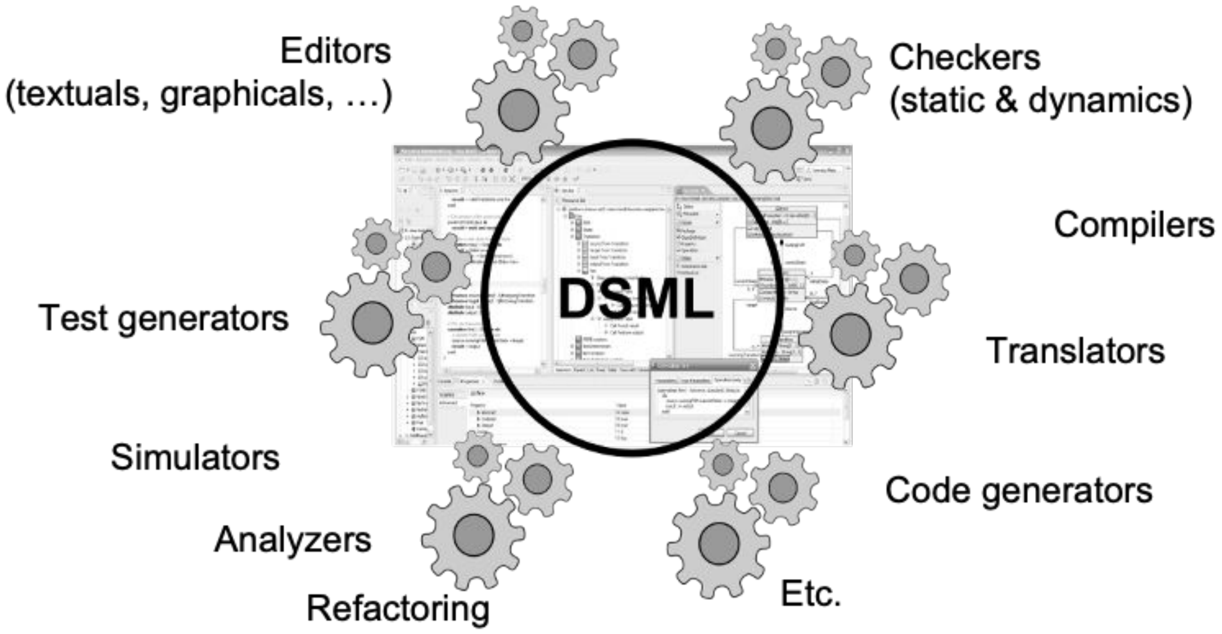
Engineering Modeling Languages: Turning Domain Knowledge into Tools
Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim R.H. Steel, and Didier Vojtisek
Chapman and Hall/CRC, pp.398, 2016. Companion website: http://mdebook.irisa.fr

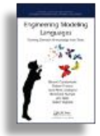
Model-Driven Engineering



Engineering Modeling Languages: Turning Domain Knowledge into Tools
Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim R.H. Steel, and Didier Vojtisek
Chapman and Hall/CRC, pp.398, 2016. Companion website: http://mdebook.irisa.fr

Model-Driven Engineering





Engineering Modeling Languages: Turning Domain Knowledge into Tools
Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim R.H. Steel, and Didier Vojtisek
Chapman and Hall/CRC, pp.398, 2016. Companion website: http://mdebook.irisa.fr

"Software languages are software too"

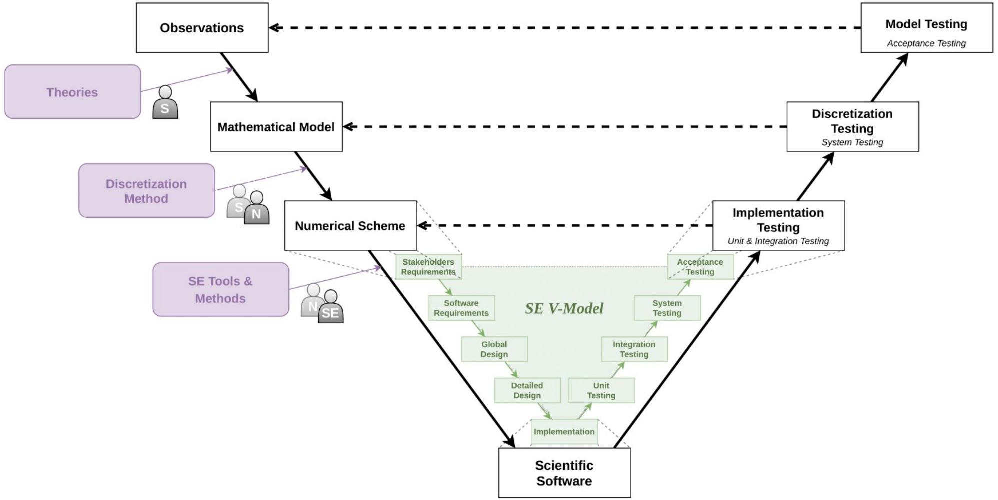
Empirical language analysis in software linguistics
J-M. Favre, D. Gasevic, R. Lämmel, and E. Pek. SLE 2011

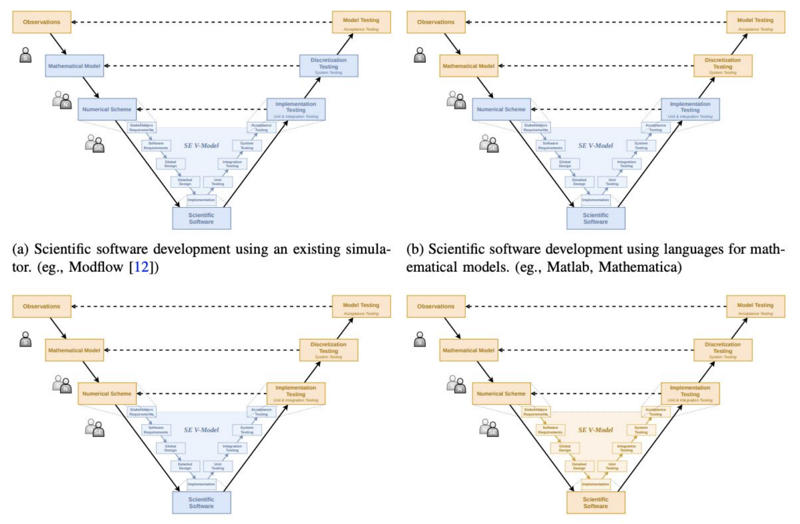
Software Language Engineering

- Application of systematic, disciplined, and measurable approaches to the development, deployment, use, and maintenance of software languages
- Supported by various kind of "**language workbench**"
  - Eclipse EMF, Xtext, Sirius, Melange, GEMOC, Papyrus
  - Jetbrain's MPS
  - Spoofax
  - MS DSL Tools
  - etc.
- Various shapes and ways to implement software languages
  - External, internal or embedded DSLs, Profile, etc.
  - Grammar, metamodel, ontology, etc.
- More and more literature, a dedicated Intl. conference (ACM SLE, cf. http://www.sleconf.org)...

Software Languages in Scientific Computing

The more general-purpose the language is the more flexibility it will provide, but also the more rigorous engineering principles and V&V activities it will require from the language user





(a) Scientific software development using an existing simulator. (eg., Modflow [12])

(b) Scientific software development using languages for mathematical models. (eg., Matlab, Mathematica)

(c) Scientific software development using languages for numerical schemes. (eg., Blitz++, SciPy, NabLab)

(d) Scientific software development using languages with system abstractions. (eg., C++, Python)

When Scientific Software Meets Software Engineering

D. Leroy, J. Sallou, J. Bourcier and B. Combemale. Computer, vol. 54, no. 12, pp. 60-71, 2021.

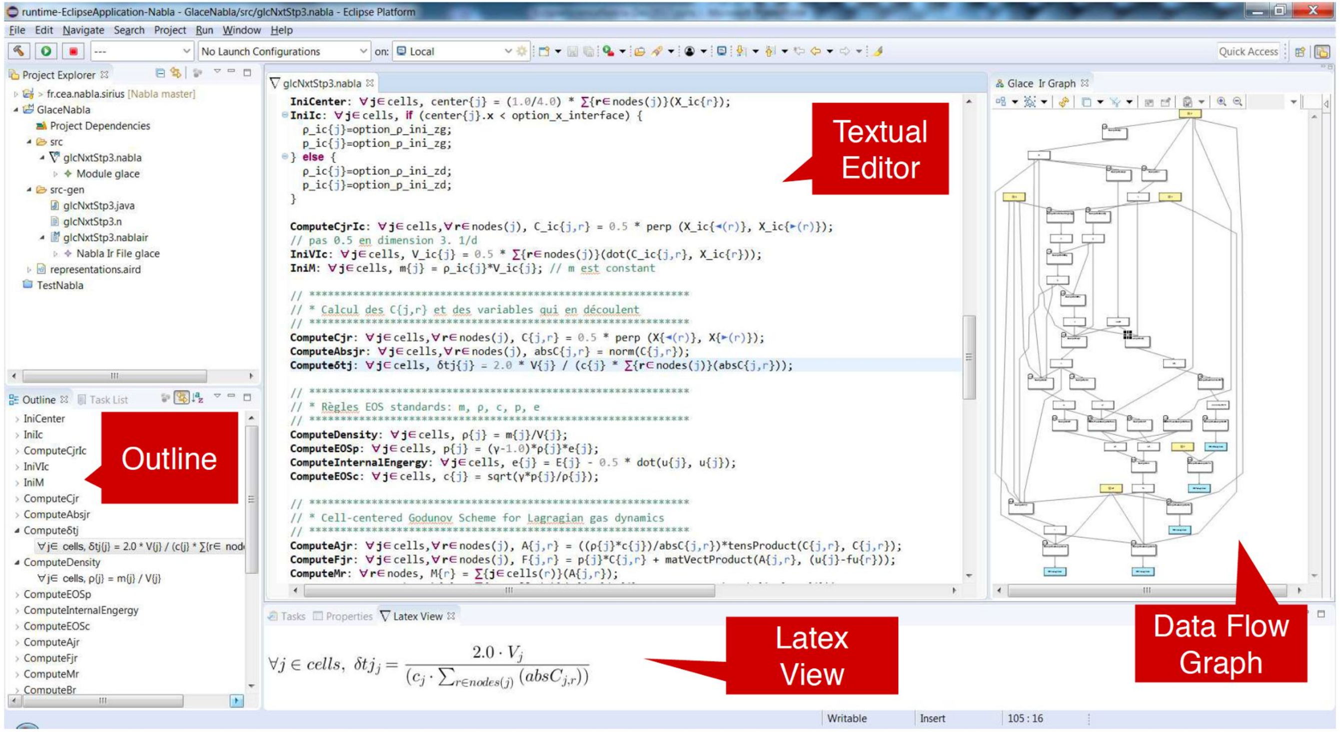
Preprint: https://hal.inria.fr/hal-03318348v1

Software Languages in Scientific Computing

Language	Mathematical Model	Numerical Scheme	Scientific Software
Mathematica (Wolfram Language)	+++	++	
MATLAB	++	++	
R	+	+	
NabLab		+++	
Julia		++	+
SciPy		++	+
Python			+
Java			++
C/C++			+++
Fortran		++	+++

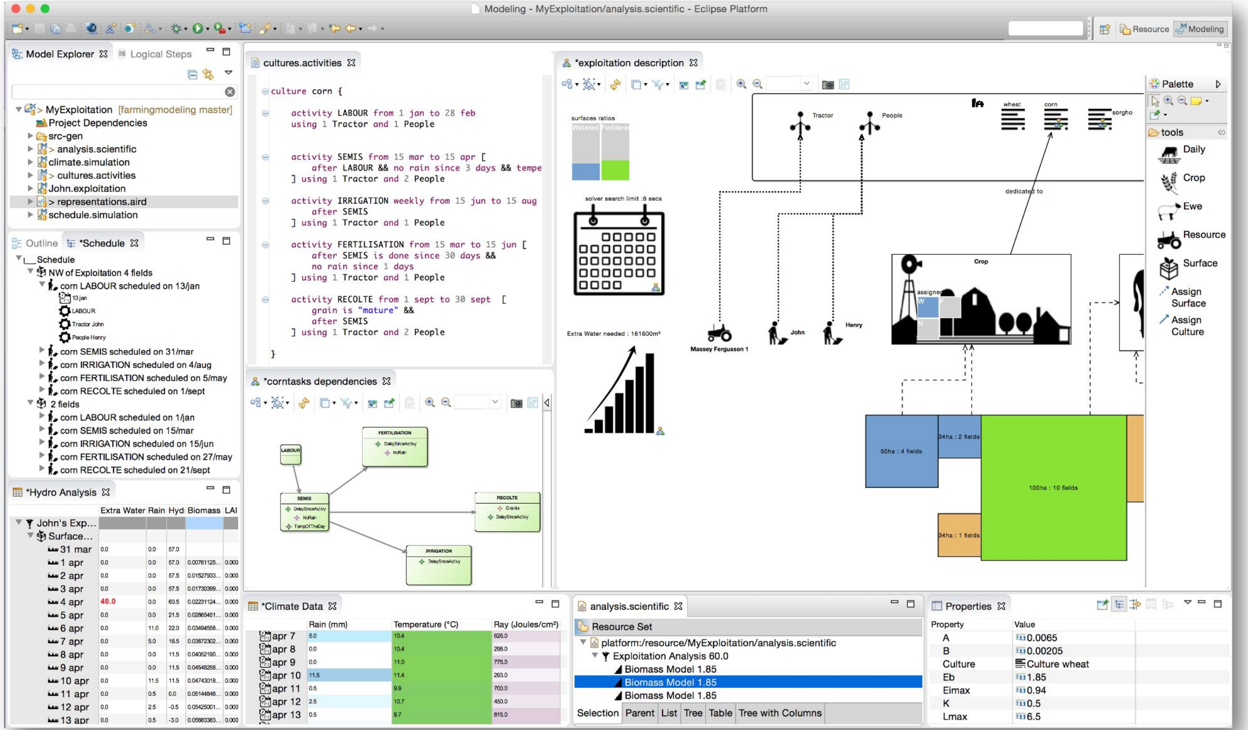
When Scientific Software Meets Software Engineering
D. Leroy, J. Sallou, J. Bourcier and B. Combemale. Computer, vol. 54, no. 12, pp. 60-71, 2021.
Preprint: https://hal.inria.fr/hal-03318348v1

Domain-Specific Languages: NabLab



Fostering metamodels and grammars within a dedicated environment for HPC: the NabLab environment (tool demo).
Benoît Lelandais, Marie-Pierre Oudot, Benoît Combemale. SLE 2018: 200-204

Domain-Specific Languages: “*DSL de Vache*”



MDE in Practice for Computational Science

Jean-Michel Bruel, Benoit Combemale, Ileana Ober, Hélène Raynal. ICCS, 2015.

Preprint: https://hal.inria.fr/hal-01141393

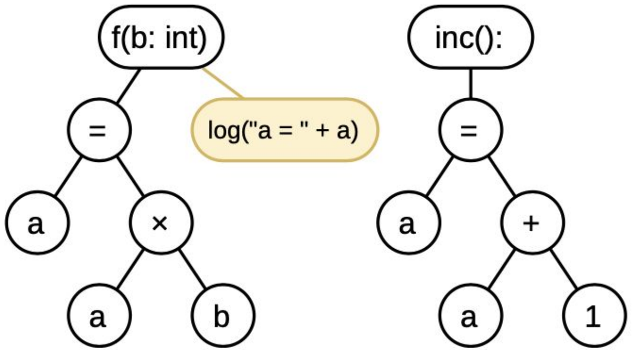
On Scientific Software Debugging

Debugging in Scientific Computing

- Common debugging facilities (i.e., step-by-step execution) are not suitable for numerical schemes with highly iterative processing
- Common **debug use cases** in scientific computing:
  - Conditional breakpoints
  - Constraint checking
  - Validation rules
  - Logging of values
- These use case each revolve around **runtime monitoring** and **logging**, respectively to:
  - determine when to perform an action (e.g., format a message), and
  - communicate the result to either the user, or a component (e.g., the debugger).

Debugging in Scientific Computing

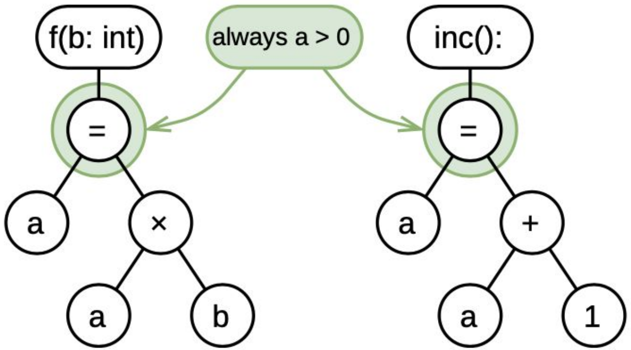
Logging:



Loggers are weaved in the AST to output context-specific messages.

- Requires **structural pointcuts**

Runtime monitoring:



Runtime monitors observe the execution to render a verdict on properties.

- Define **behavioral pointcuts**

Monilogging for Executable Domain-Specific Languages
Dorian Leroy, Benoît Lelandais, Marie-Pierre Oudot, Benoit Combemale. *SLE 2021.*

Cross-fertilization of Logging & Runtime Monitoring?

A few examples of cross-fertilization between logging and monitoring:

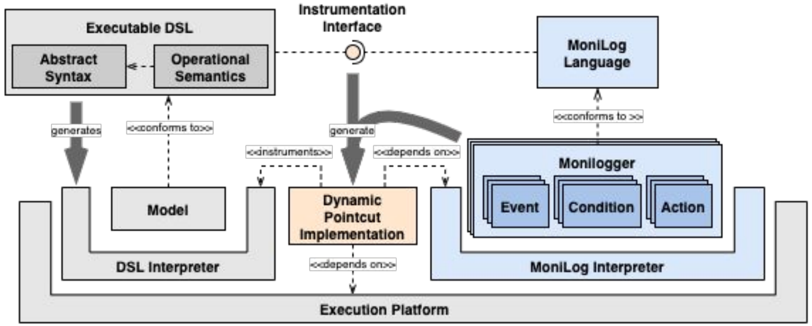
- Logging the steps in the evaluation of a temporal property
  - *e.g.*, print the normal of a shock wave while checking it eventually reverses.
  - Logging benefits monitoring.
- log values in arbitrary complex situations:
  - *e.g.*, log a message when the pressure becomes negative.
  - Monitoring benefits logging.

MoniLog (soon renamed '*SciHook*')

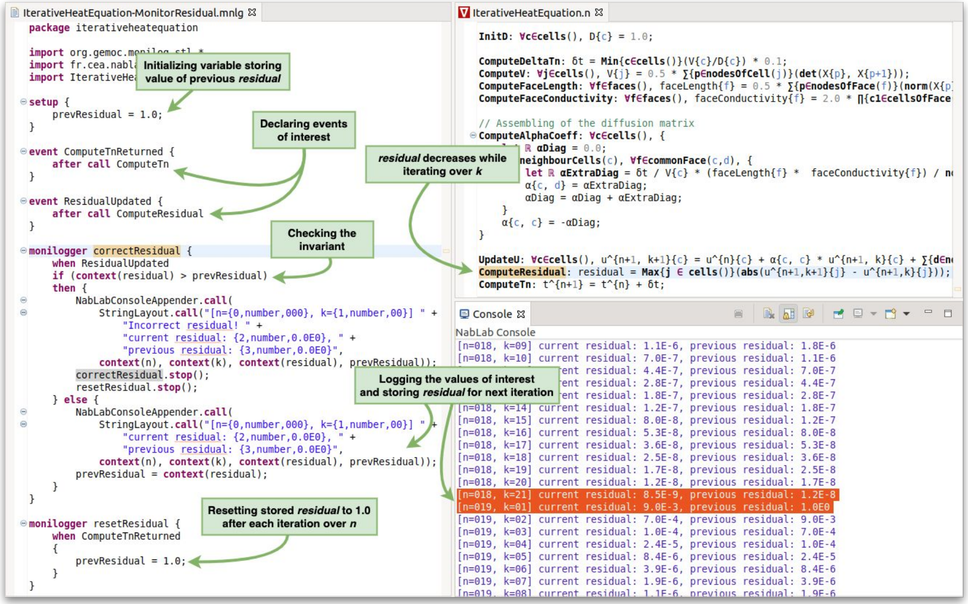
- Analyzing complex or data-intensive behaviors requires insightful data
  - alternative to debugging in scientific computing
- MoniLog: a unifying framework for defining:
  - **loggers**: extract data from program state and format it as messages
  - **runtime monitors**: evaluation of temporal properties on programs
  - **moniloggers**: combinations of loggers and monitors
- Moniloggers are defined in a language-agnostic way, relying on an instrumentation interface provided by DSLs
  - applied to any DSLs
  - keep monitoring and logging concerns out of domain concerns

Monilogging for Executable Domain-Specific Languages
Dorian Leroy, Benoît Lelandais, Marie-Pierre Oudot, Benoit Combemale. *SLE 2021.*

MoniLog for Interpreted DSLs

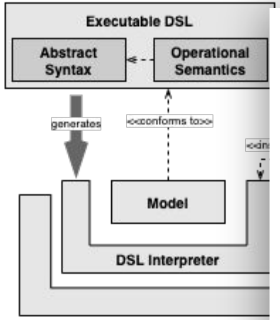


Implementation on the JVM, using either AspectJ or Truffle



Monilogging for Executable Domain-Specific Languages
Dorian Leroy, Benoît Lelandais, Marie-Pierre Oudot, Benoit Combemale. *SLE 2021.*

MoniLog for Interpreted DSLs



Instrumentation

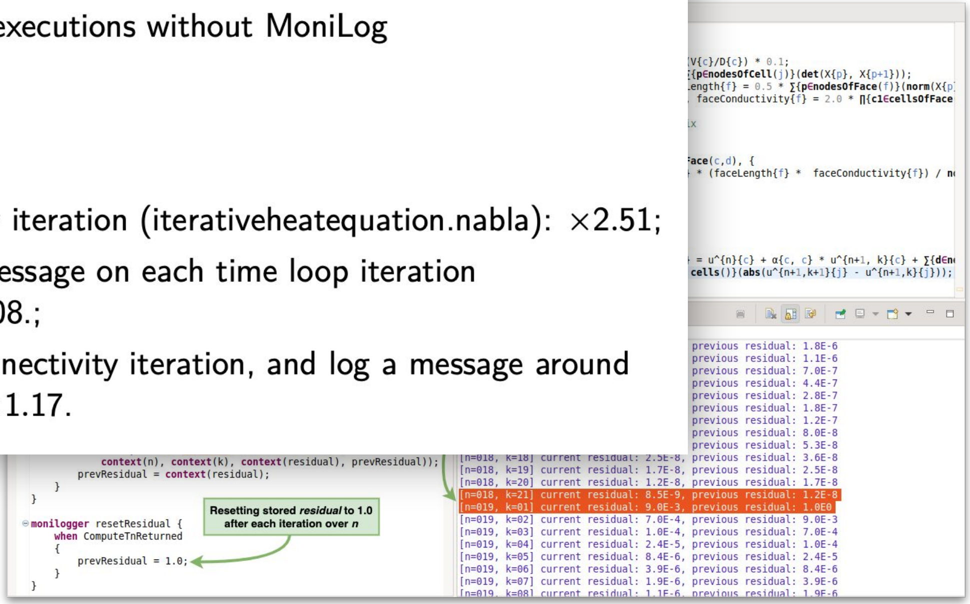
Implementation using either As

Benchmark:

- Mean execution time overhead
- over 100 executions with and 100 executions without MoniLog
- in various scenarios.

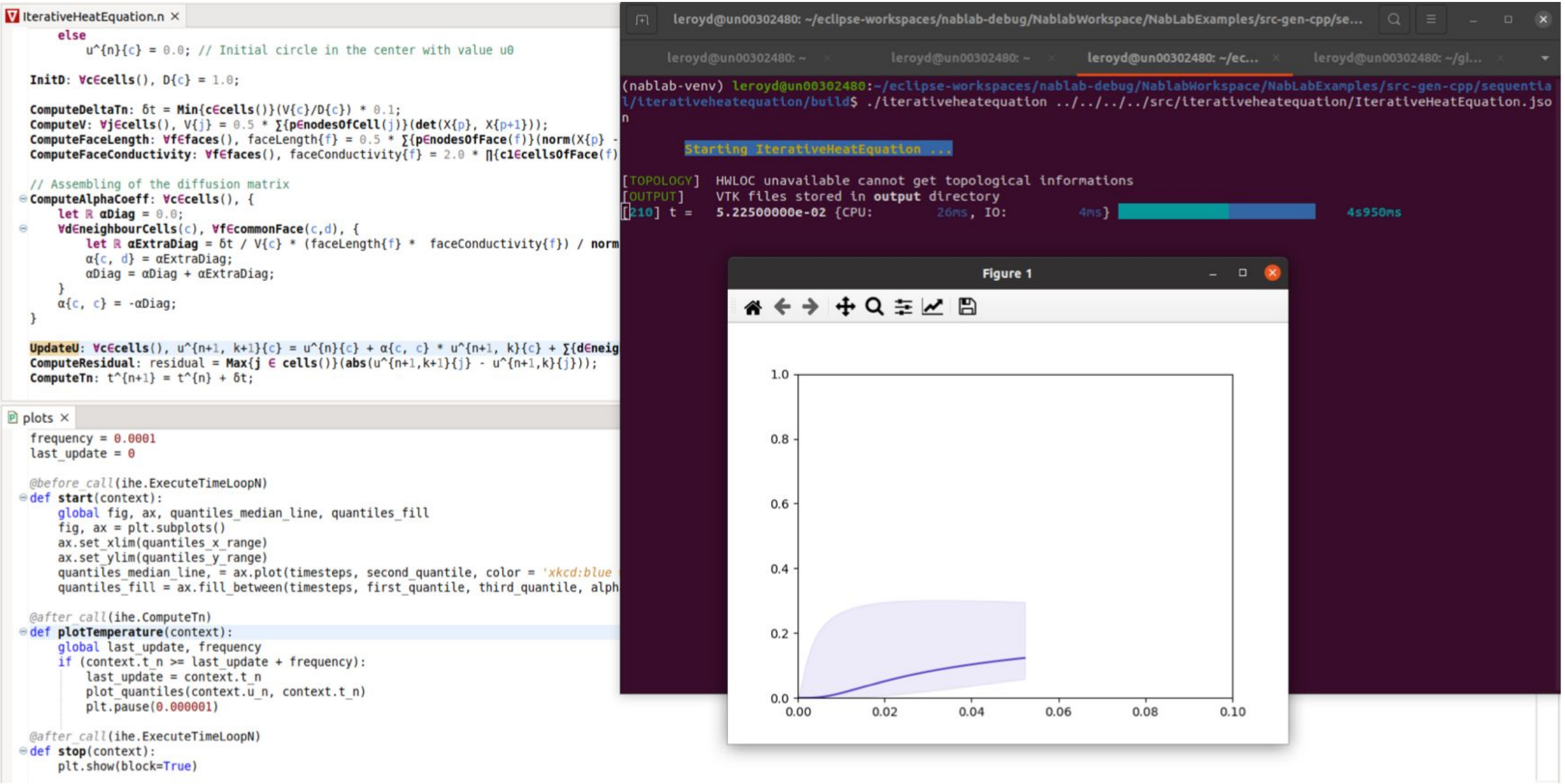
Scenarios and results:

- log a message on each connectivity iteration (iterativeheatequation.nabla): $\times 2.51$;
- evaluate 6 expressions and log a message on each time loop iteration (iterativeheatequation.nabla): $\times 1.08$.;
- evaluate an expression on each connectivity iteration, and log a message around half of the time (glace2d.nabla): $\times 1.17$.

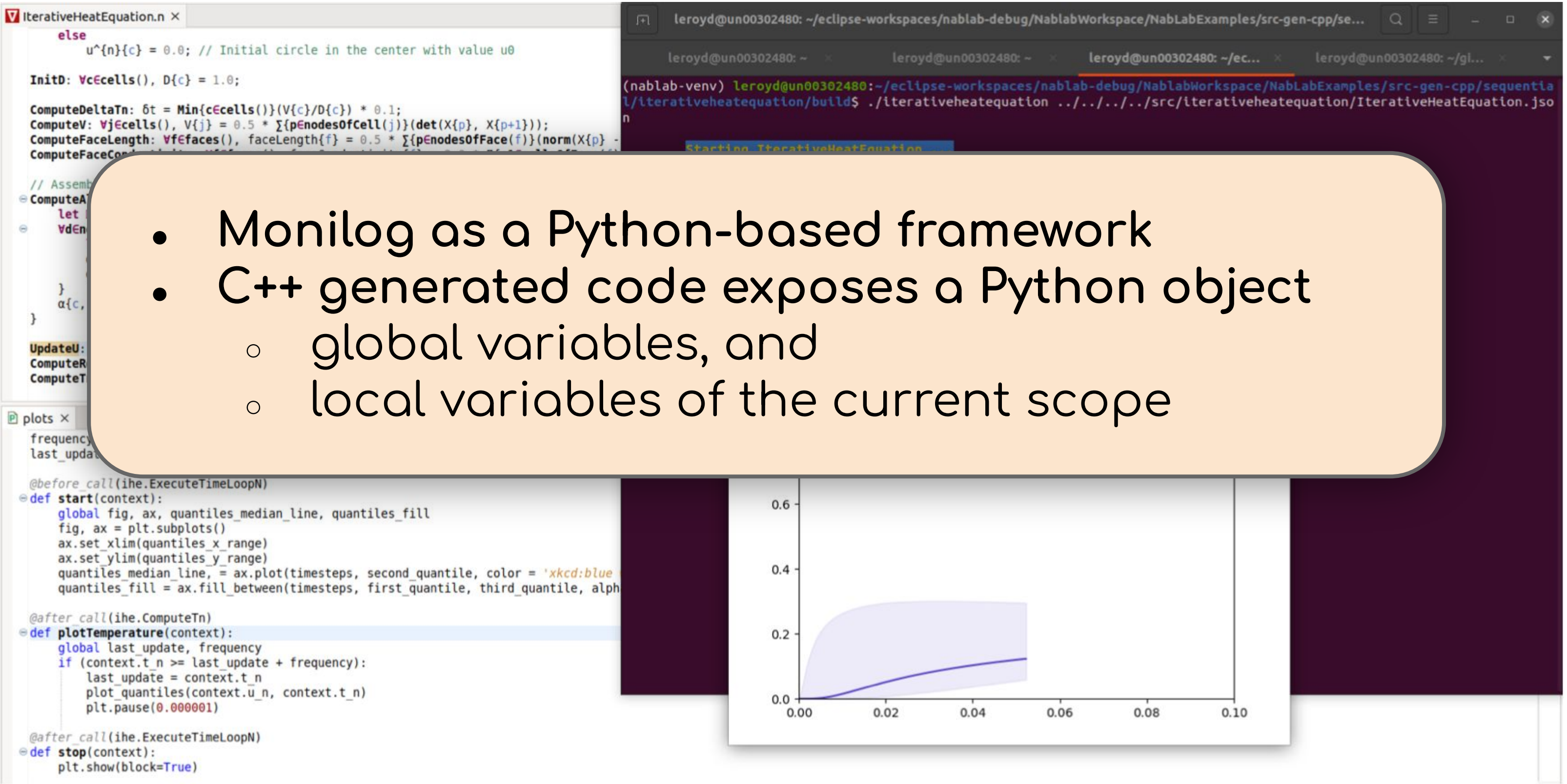


Monilogging for Executable Domain-Specific Languages
Dorian Leroy, Benoît Lelandais, Marie-Pierre Oudot, Benoit Combemale. *SLE 2021.*

MoniLog for Compiled DSLs

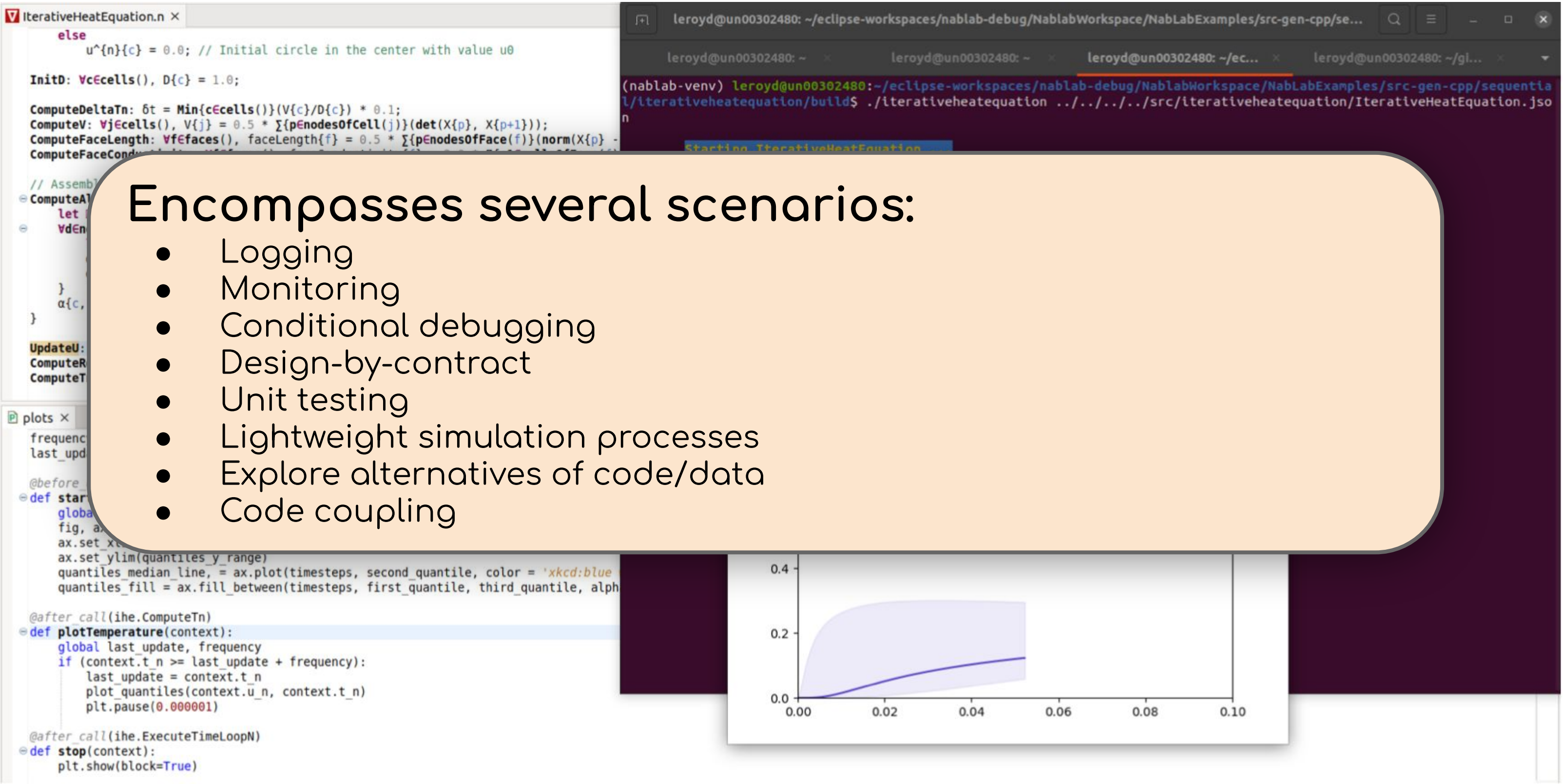


MoniLog for Compiled DSLs



- Monilog as a Python-based framework
- C++ generated code exposes a Python object
  - global variables, and
  - local variables of the current scope

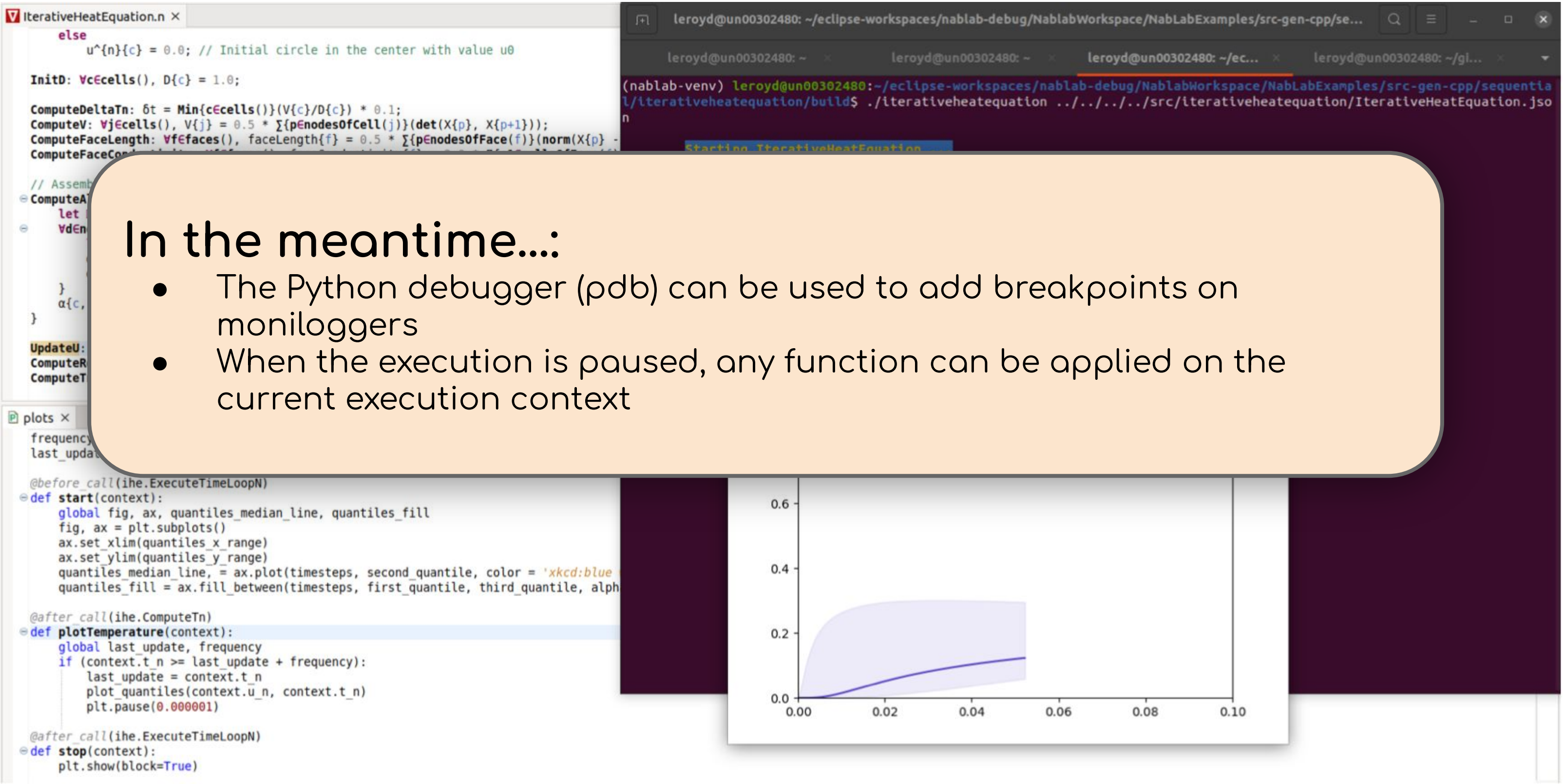
MoniLog for Compiled DSLs



Encompasses several scenarios:

- Logging
- Monitoring
- Conditional debugging
- Design-by-contract
- Unit testing
- Lightweight simulation processes
- Explore alternatives of code/data
- Code coupling

MoniLog for Compiled DSLs

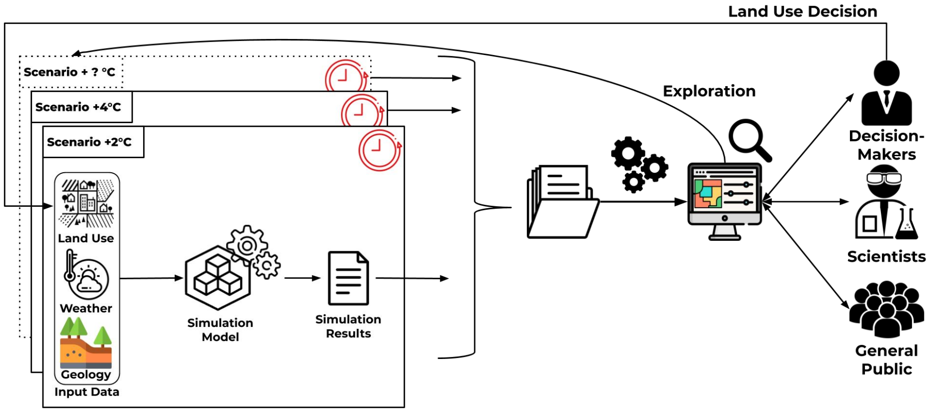


In the meantime...:

- The Python debugger (pdb) can be used to add breakpoints on moniloggers
- When the execution is paused, any function can be applied on the current execution context

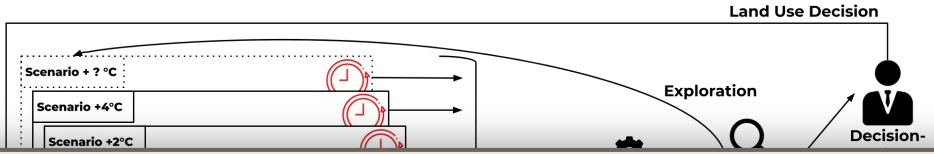
On Trade-off Analysis

Tradeoff Analysis in Scientific Computing



- Integrated environment for scientific computing
  - Flexible, agile, collaborative, distributed & adaptive
- Support for trade-off analysis and decision making
  - Exploration of scenarios
  - Integrity of projections
  - interactivity
  - Generalisation
- Application to environmental sciences
  - in collaboration with OSUR (UR1)
  - other collaborations with Lancaster University (e.g., Data Science of the Natural Environment)

Tradeoff Analysis in Scientific Computing



How to tailor scientific software to support decision-making?

- Integ
  -
- Support for trade-off analysis and decision making
  - Exploration of scenarios
  - Integrity of projections
  - interactivity
  - Generalisation
- Application to environmental sciences
  - in collaboration with OSUR (UR1)
  - other collaborations with Lancaster University (e.g., Data Science of the Natural Environment)

Tradeoff Analysis in Scientific Computing

High Performance Computing (HPC)

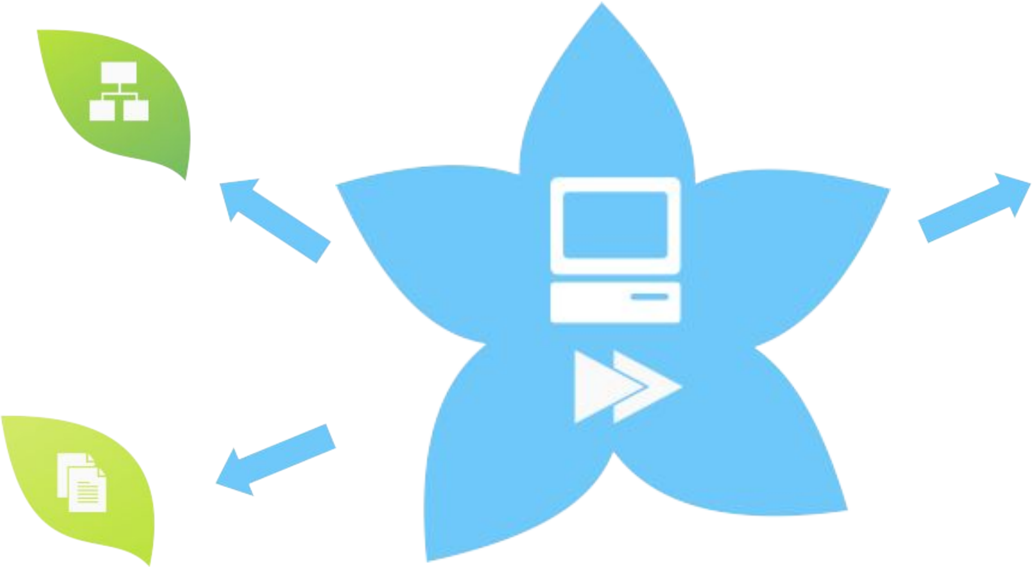
Increase and improve the computational hardware and infrastructure

Model Reduction

Simplify the mathematical model

Data-driven Approach

Build an analytic model



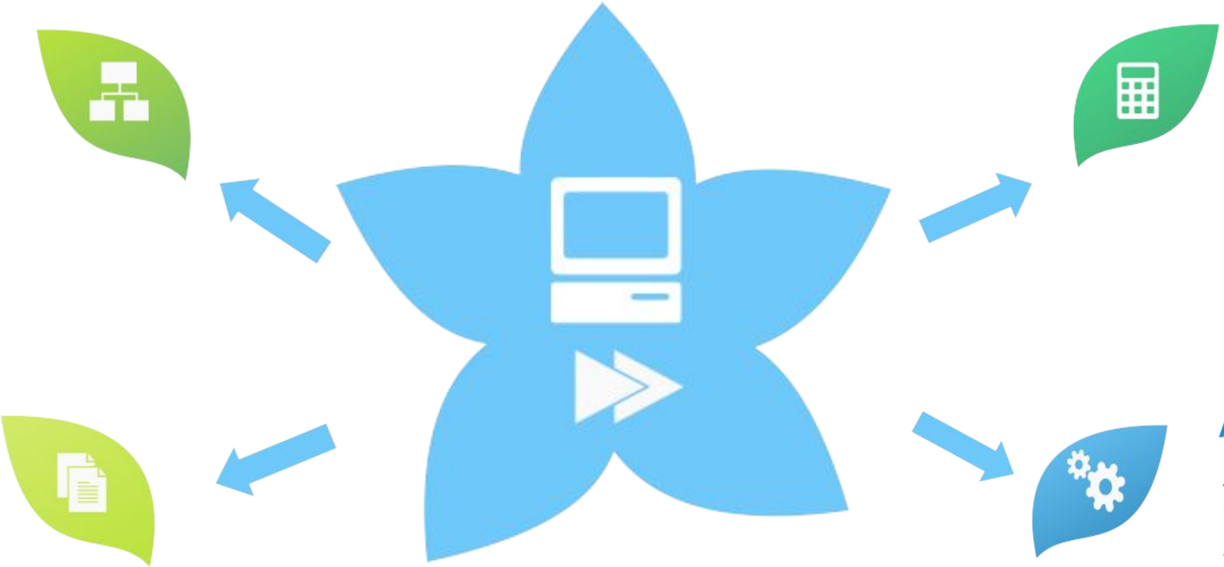
Tradeoff Analysis in Scientific Computing

High Performance Computing (HPC)

Increase and improve the computational hardware and infrastructure

Data-driven Approach

Build an analytic model



Model Reduction

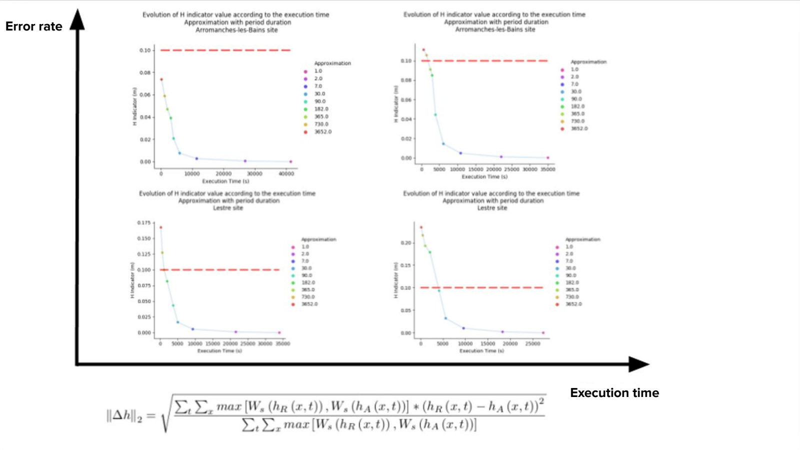
Simplify the mathematical model

Approximate Computing

Approximate the computation. Trade-off between accuracy and time execution

Approximate Scientific Computing

- Reduce the simulation time to better support trade-off analysis and decision making
- Application of approximate computing to scientific computing
- Work on the simulation code (white box) or the input data (black box)
- Require a domain-specific interpolation operator
- Challenge: infer the approximation bound



Loop Aggregation for Approximate Scientific Computing
June Sallou, Alexandre Gauvain, Johann Bourcier, Benoît Combemale, Jean-Raynald de Dreuzy. *ICCS 2020: 141-155.*

SE4Science:
Call For Actions!

SE4Science: Opportunities

- **Increasing demand**
  - Science-driven society
  - Acceleration of natural phenomena
  - Breakthroughs in engineering required innovative thinking
- **Technology evolution**
  - Continuous evolution on computing capabilities
  - Progresses on M&S tools and methods
  - Breakthroughs on development environments (e.g., language servers)
  - Cloud infrastructure and web-based technologies as key enablers

SE4Science: Objectives

- **International leadership** within the scientific community
  - proper programming foundations
  - tools and methods for collaborative, effective and reliable development

- **Competitiveness** of the French industry

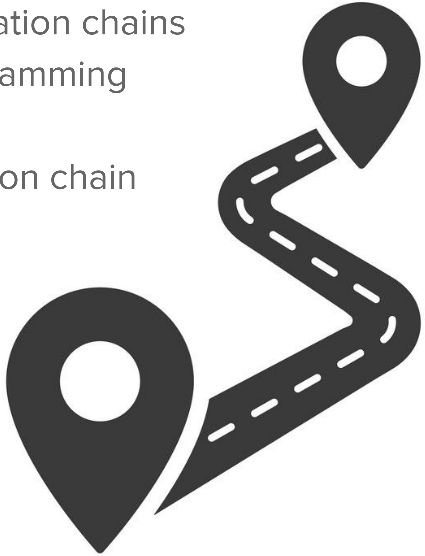
- **Broader engagement** with decision and policy makers, and the general public

SE4Science: Roadmap

- Programming Foundations
- Scientific Software Validation and Verification
- AI-enhanced Scientific Computing
- Scientific Computing Virtual Lab

SE4Science: Roadmap

- **Programming Foundations**
    - Domain-Specific Languages for simulation processes, and compilation chains
    - Sound combination of literate, exploratory, live and polyglot programming
    - Language constructs for heterogeneous model coupling
    - Variability management for input data, source code, and compilation chain
    - Reproducibility and deep variability management
- Scientific Software Validation and Verification
- AI-enhanced Scientific Computing
- Scientific Computing Virtual Lab



SE4Science: Roadmap

- Programming Foundations
- **Scientific Software Validation and Verification**
  - Testing framework and design-by-contract
  - Tradeoff with privacy
  - Experimental frame and correctness envelope for scientific models
  - Compiler V&V
- AI-enhanced Scientific Computing
- Scientific Computing Virtual Lab

SE4Science: Roadmap

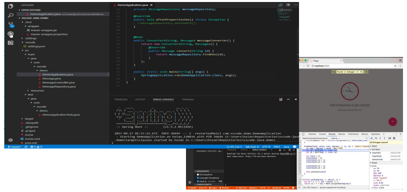
- Programming Foundations
- Scientific Software Validation and Verification
- **AI-enhanced Scientific Computing**
  - AI-based simulation
  - Predictive models for design-space exploration
  - AI-based assistants/recommenders and tradeoff analysis tools
    - for scientific software development
    - for model calibration, compilation/execution parameters
    - for deployment and delivery
  - Transfer learning for reuse
  - Multi-fidelity and frugality
- Scientific Computing Virtual Lab

SE4Science: Roadmap

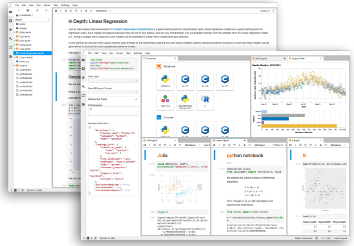
- Programming Foundations
- Scientific Software Validation and Verification
- AI-enhanced Scientific Computing
- **Scientific Computing Virtual Lab**
    - Programming interface (e.g., computational notebooks)
    - Web-based, cloud-native, collaborative and distributed IDE
    - Digital twin framework

Virtual Lab

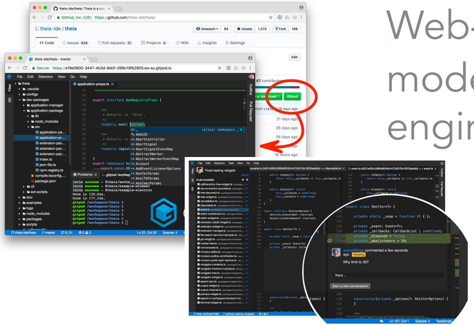
Socio-technical coordination



Lightweight, modular, customizable, distributed and self-adaptable scientific and engineering platforms…



Polyglot, literate programming



Web-based, Collaborative modeling, modeling flow, social engineering

Exploratory and live programming, digital twin

SE4Science: Roadmap

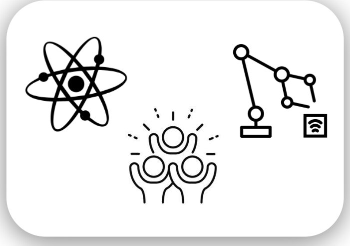
- **Programming Foundations**
  - Domain-Specific Languages for simulation processes and compilation chains
  - Sound combination of literate, exploratory, live and polyglot programming
  - Language constructs for heterogeneous model coupling
  - Variability management for input data, source code, and compilation chain
  - Reproducibility and deep variability management
- **Scientific Software Validation and Verification**
  - Testing framework and design-by-contract
  - Tradeoff with privacy
  - Experimental frame and correctness envelope for scientific models
  - Compiler V&V
- **AI-enhanced Scientific Computing**
  - AI-based simulation
  - Predictive models for design-space exploration
  - AI-based assistants/recommenders and tradeoff analysis tools
  - Transfer learning for reuse
  - Multi-fidelity and frugality
- **Scientific Computing Virtual Lab**
  - Programming interface (e.g., computational notebooks)
  - Web-based, cloud-native, collaborative and distributed IDE
  - Digital twin framework

SE4Science: Expected Impact

- Establish **software engineering principles** for engineering scientific software
- Consolidate research activities into **open-source tools** accessible to scientists, engineers, decision makers and the general public
- Establish a strong **leadership** in the scientific, technological and industrial communities

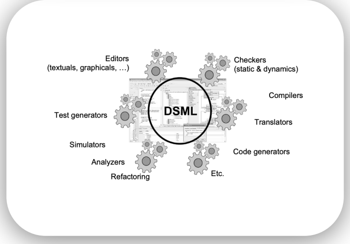
SE4Science: Expected Impact

Take Away Messages



Scientific Computing

- In science, engineering and society
- Complex and specific SDLC
- Requires proper software engineering principles



(Domain-Specific) *Software* Languages

- Domain-specific abstractions
- Support for coordination, debugging, approximation…

SE4Science: a **vision** with associated **roadmap**

- Timely and relevant
- Impactful
- Unique opportunity for the French community