| galois |

Generative AI for Rigorous Digital Engineering

Joe Kiniry
Galois

May 2024
HCSS 2024

Rigorous Digital Engineering (RDE) is an engineering process and methodology created and refined over the past two decades by this author, first during his days as a Professor, and over the past decade while a Principal Scientist at Galois. RDE combines the practical use of Model-Based Engineering (MBE) with Applied Formal Methods (AFM) in such a way that formality is hidden from engineers using a technique called Secret Ninja Formal Methods (SNFM). RDE has been demonstrated on hundreds of software and hardware systems, and has been used on dozens of Galois projects to create secure, high-assurance software systems and ASICs. The kinds of engineering that have been incorporated into RDE include systems, software, firmware, hardware, safety, and security engineering.

Galois has been experimenting with using untrustworthy AI as partners in nearly every facet of Rigorous Digital Engineering. The facets include doing domain engineering, requirements engineering, security engineering, systems engineering, model-based engineering and design at the source code level, product line engineering, software engineering and hardware engineering, building various kinds of assurance cases involving both runtime verification and testing for validation, and formal verification.

DARPA Space-BACN recently funded the development of an entire course on Rigorous Digital Engineering. The first edition of the course is now complete, and is initially being given internally at Galois. It is meant to be publicly available to the world this year. While the course comes with many historic example RDE projects (e.g., SHAVE and HARDENS), a wholly new RDE Course Demonstrator called **ENGRAVE** is being created by this author. ENGRAVE is an example RDE system that focuses on the design, development, and assurance of a domain-specific language (DSL) and architecture (DSA), including a family of programming and specification languages, and a full small development tool suite including a type checker, interpreter, testbench generator, and a formal verification tool.

This kind of case study is important because many modern DoD/DIB systems are beginning to use DSLs and DSAs more commonly, and thus including these kinds of components in the Dataset improves its utility. Also, ENGRAVE resembles many of the technologies that Galois creates in its R&D. Finally, the ENGRAVE project is one of the main places where we are experimenting with the use of modern AIs.

In only a few days of work, leveraging generative AIs and prototype LLM used as described above, I have already written: (0) the specific RDE process and methodology for the ENGRAVE project, (1) a CONOPS for the system, (2) a domain engineering model in the Lando system specification language, (3) system requirements, also in Lando, (4) designed the ENGRAVE family of languages, including its grammar and informal semantics, (5) a product line model describing the family of languages, command line options, CI/CD/CV system, and assurance case, (6) a system design sketch in UML and SysMLv2, (7) a set of grammars for the full language family, (8) the entire front-end of the tool flow including a lexer, parser, and intermediate representation, (9) a full container environment for CI, CD, and CV, (10) instantiated that CI/CD/CV environment with continuously measured metrics of system completeness and correctness, and (11) instantiated a set of standard developer environments using popular IDEs and command-line tools. Prior to HCSS taking place, I expect to complete the ENGRAVE demonstrator.

My HCSS talk will review modern AI as applied to RDE by walking through the ENGRAVE project.

|galois|

Advancing computer science R&D
Creating trustworthiness in critical systems

Computer Security - Correctness - Cryptography
Cyber-Physical Systems - Data Science - Formal Methods
Human-Computer Interaction - Languages - Machine Learning
Rigorous Digital Engineering - Semiconductors

Clients

AWS	IARPA	NRC
DARPA	NASA	NSA
DHS	NIH	NSF
DOD	NIST	SDA
DOE		

Offices

Portland, OR
Dayton, OH
Arlington, VA
Minneapolis, MN

History

Founded in 1999
120+ employees

Spin-outs

Tangram Flex
Tozny
Free & Fair
Niobium Microsystems

A different kind of company

No managers
We have no fixed hierarchy of rigid positions and titles, and no traditional managers.

Radical transparency
Everything is transparent by default: company financials, decision making, open meetings, and even salaries.

Choose your work
Research engineers choose the projects they work on, and move freely between projects depending on personal interests and career goals.

Ownership
Employees own the company together, making important decisions as a group and partaking in the financial success of the company.

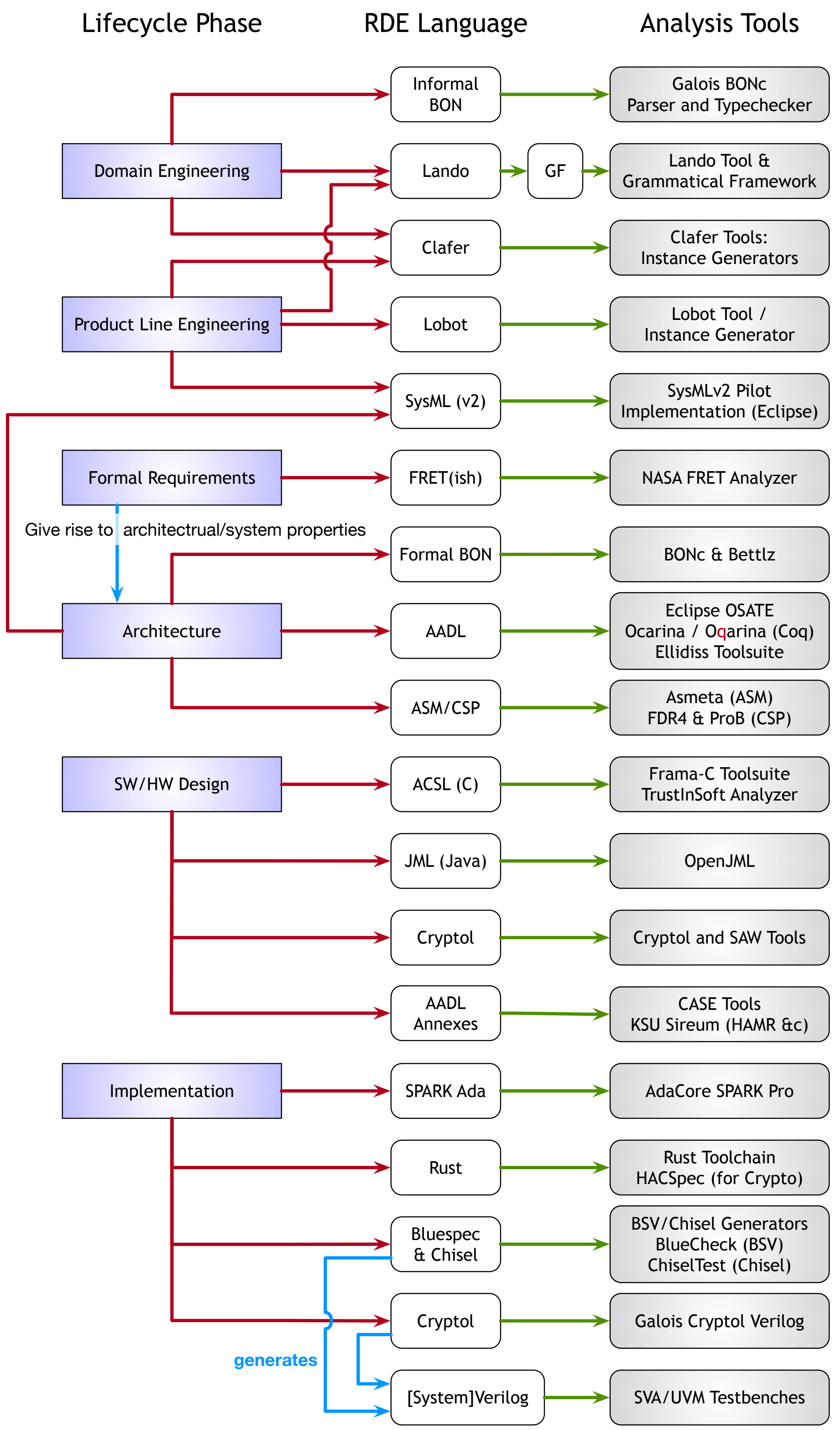
More info at lifeatgalois.com

The HCSS Arc of RDE

- HCSS 2015: *Verifying Elections (poster)*
  - Free & Fair: RDE for nationally critical infrastructure—e.g., Microsoft ElectionGuard, CORLA, BVS 2019
- HCSS 2019: *Galois Does Secure Hardware*
  - creating correct-by-construction ASICs (and, later, FPGAs) that include RISC-V CPUs and crypto
- HCSS 2022: *High Assurance Rigorous Digital Engineering for Nuclear Safety (HARDENS)*
  - creating correct-by-construction, safety-critical DI&C systems with formally assured SW, FW, & HW
- HCSS 2024: *Generative AI for Rigorous Digital Engineering*
  - democratizing RDE by leveraging untrustworthy LLMs

Rigorous Digital Engineering

- RDE is all about how to use applied formal methods concepts, tools, and technologies in practical, high-impact ways leveraging compositionality via refinement and equivalence relations
- RDE supports all forms of digital engineering: domain, requirements, software, firmware, hardware, systems, safety, and security engineering
- different problems require different tools—that's a lot to learn! So how might we democratize RDE?



Lifecycle Phase
RDE Language
Analysis Tools
Informal BON
Galois BONc Parser and Typechecker
Domain Engineering
Lando
GF
Lando Tool & Grammatical Framework
Clafer
Clafer Tools: Instance Generators
Product Line Engineering
Lobot
Lobot Tool / Instance Generator
SysML (v2)
SysMLv2 Pilot Implementation (Eclipse)
Formal Requirements
FRET(ish)
NASA FRET Analyzer
Give rise to architectrual/system properties
Formal BON
BONc & Bettlz
Architecture
AADL
Eclipse OSATE Ocarina / Oqarina (Coq) Ellidiss Toolsuite
ASM/CSP
Asmeta (ASM) FDR4 & ProB (CSP)
SW/HW Design
ACSL (C)
Frama-C Toolsuite TrustInSoft Analyzer
JML (Java)
OpenJML
Cryptol
Cryptol and SAW Tools
AADL Annexes
CASE Tools KSU Sireum (HAMR &c)
Implementation
SPARK Ada
AdaCore SPARK Pro
Rust
Rust Toolchain HACSpec (for Crypto)
Bluespec & Chisel
BSV/Chisel Generators BlueCheck (BSV) ChiselTest (Chisel)
Cryptol
Galois Cryptol Verilog
generates
[System]Verilog
SVA/UVM Testbenches

AI in RDE

- “classical” AI has always been a tool in our toolbox
  - NNs, symbolic AI, knowledge representation, multi-agent systems, decision trees, etc.
  - solvers that leverage AI felt no different than others we use for constraint solving, SAT, SMT
- sitting behind RDE is *kind theory*, developed in 1999–2002, which relies upon an auto-epistemic non-classical logic
  - auto-epistemic: permits one to contextualize formalizations of knowledge “per user”
  - non-classical: permits one to reason in the presence of underspecification and contradictions

=> we were well situated to contend with hallucinating, unpredictable black-box “generating” and “reasoning” LLMs

Generative AI Burst on the Scene

- we had friends to go DeepMind in the early 2010s and hint that very interesting things were afoot
- we heard about OpenAI for the first time in ~2016
- 2018 GPT burst on the scene with BERT and GPT-1
  - GPT used to mean the Gabidulin-Paramonov-Tretyakov cryptosystem or Geometry Proof Tutor
  - started to get us curious about new kinds of NLP
  - we had been using type-logical grammars/semantics since circa 2000 as it connected to *kind theory*
- in 2021/22 suddenly we started seeing remarkable, surprising results in image generation and new NLP
- Galois started leaning in harder with IR&D into learning, hiring, and experimentation in 2022

Flavors and Challenges of LLMs

- we are hardcore open source / hardware geeks
- but all of the strongest LLMs are commercial, proprietary, and exhibit problems from an IP POV, so disappointed
- partly mitigated by the concurrent explosion of open models, but the effort is scattered, frantic, and induced enormous first-to-publish competition
  - impossible to keep up with multiple arXiv papers a day
- we cannot perform R&D in developing open models
  - we cannot afford experts—they all take the $$$
  - we cannot afford the training—Gov-funded SMEs do not receive millions just to train a model
- we wish to use LLMs on proprietary, and perhaps even classified data, models, and systems
- => how to perform R&D on and with LLMs as they move forward?

General Shape of Experiments

- The Big Question: *How to use an untrustworthy blackbox to build trustworthy systems?*
- questions to answer about LLM interactions:
  - How to maintain context given limited memory?
  - How to help help an LLM find consistency and stay on task?
  - How to automatically identify hallucinations?
  - How to contextualize answers for the user?
  - How to ensure outputs are mechanized?
  - How to mitigate the fact that LLMs are purely syntactic?

An LLM-neutral Architecture

- these questions lead to a design an LLM-neutral architecture
  - mitigates how quickly the world is moving
  - embeds our answers to these questions
  - based on a (prelude, template, prompt) triple for I/O

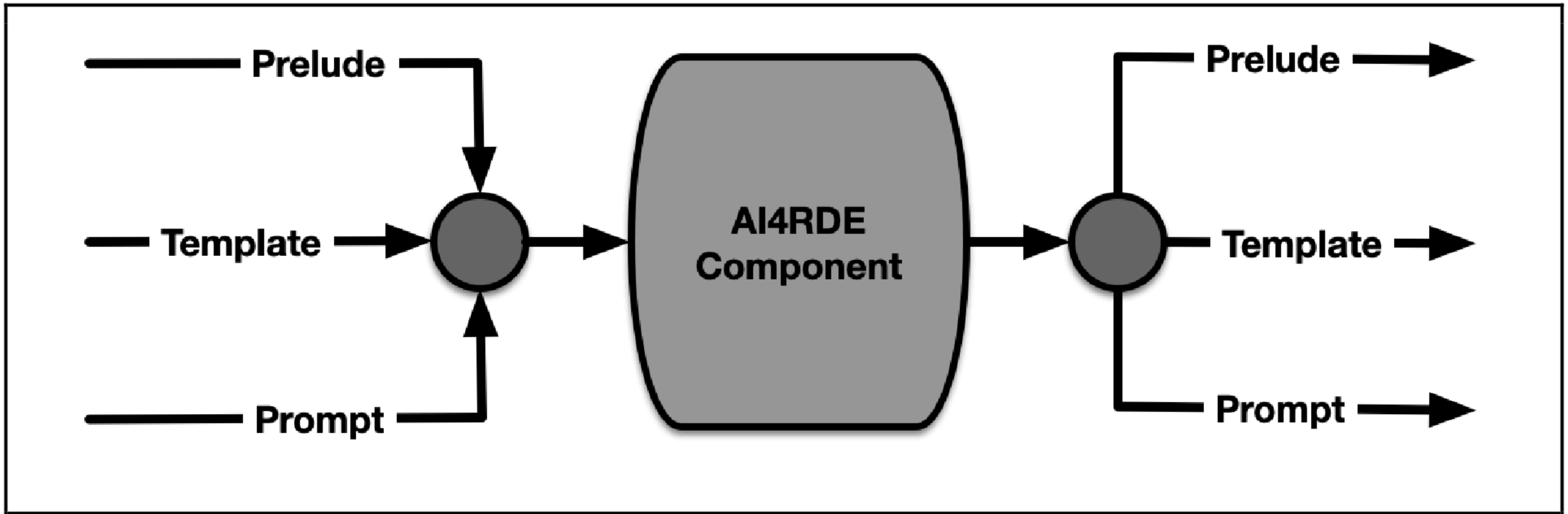


Figure 2: ***The Architecture Specification of an AI4RDE Tool Suite Component***

Key Aspects of Architecture

- realize mechanization turns syntax into semantics
- composability into pipelines & loops with fixpoints
- components as generators or evaluators
- facilitates user-centric design

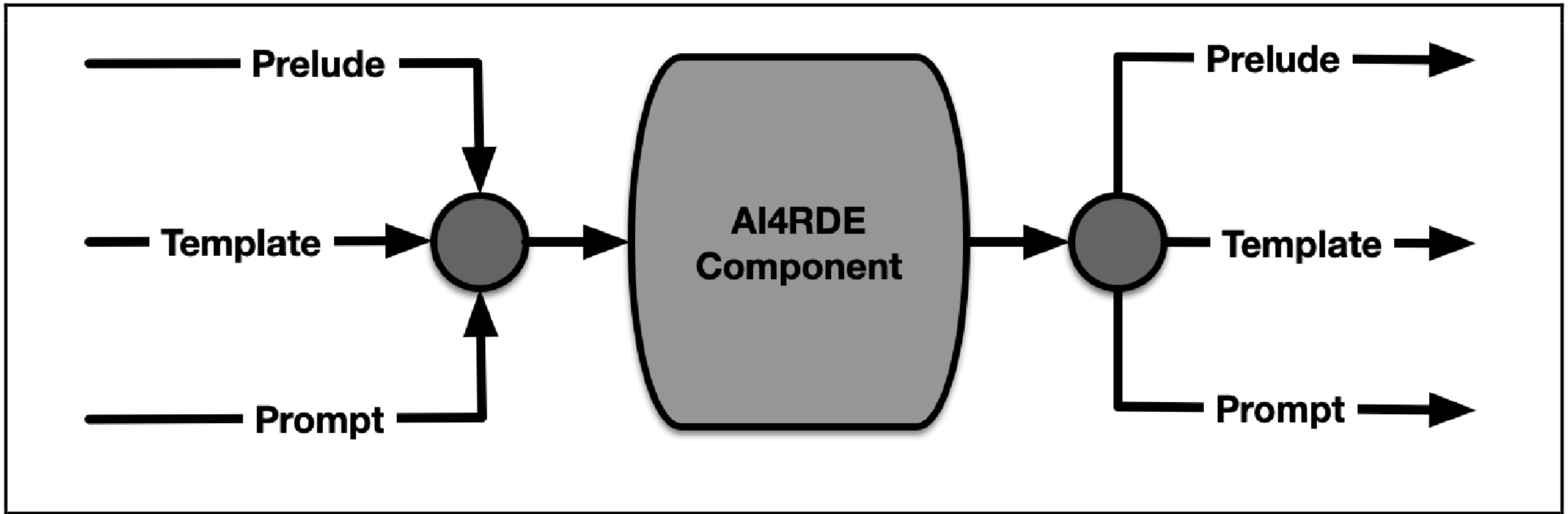


Figure 2: *The Architecture Specification of an AI4RDE Tool Suite Component*

User-centric Generative AI

- the "user" of an AI4RDE component is either a person or another AI4RDE component
- use a user-centric design methodology to develop UI/UX concepts—we cannot "bolt on" UX later
- => contextual user experience for input and output

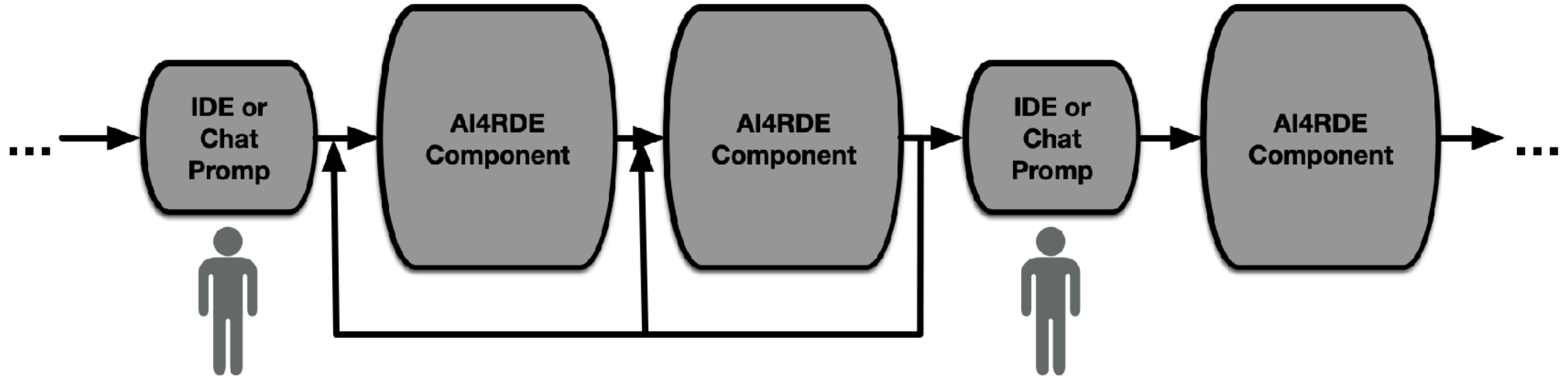


Figure 3: The Architecture Specification of the AI4RDE Tool

A Sample of Many Experiments

- co-writing CONOPs and semi-formal and formal domain engineering models (Lando, Alloy, Coq, &c)
- co-writing semi-formal and formal requirements, evolving Lando requirements into FRET properties
- co-writing formal architectural, protocol, algorithm, and behavioral interface specifications
- visual input for graphical formalizations (ASMs, AADL, SysML, UML diagrams, git trees, &c)
- visual & textual input for grammars and PLE
- automatic generation of test cases based upon formal grammars and type systems

What’s Next

- a short DARPA-funded project diving into this IR&D with relish and ambition will start soon (GAI4RDE)
- an already-written DARPA-funded RDE course—including bibliography/reading list, homework assignments, and a demonstrator system—is being given and recorded at Galois to share with all
- the course demonstrator project, called ENGRAVE (ENgineering GRAmmars and Verification Environments), is being built “by hand” and while using generative AI
  - it and HARDENS (HCSS 2022) provide two open source experimental frames for GAI4RDE
  - ENGRAVE resembles the tools Galois creates everyday: it is a new simple-but-non-trivial family of programming languages and associated tooling (all of the way from lexer to formal verification)
  - we are witnessing *significant* efficacy using generative AI