

TENON

INTEGRATION DESIGN & VERIFICATION TOOL SUITE



Know your integration works before the lab tells you it doesn't.

Tenon is an engineering tool suite that lets your engineers define how the software in your system should fit together, then proves that it does. It checks the design for problems, tests the software your team or vendors actually built against it, and traces every result back to your requirements. It runs in the terminal and in CI, for engineers and the AI agents working alongside them.

THE PROBLEM

Most software faults are designed in early and found late.

90%

of software faults are introduced during requirements, design and coding

80%

aren't found until after unit test

1 in 5

isn't found until the system is in operation, where it costs much more to remove

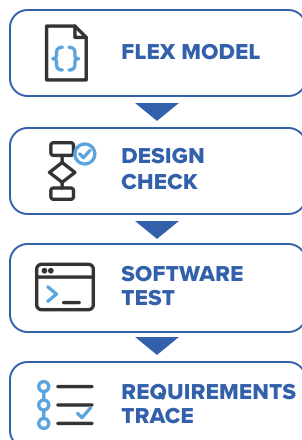
AI is widening that gap. Coding agents now write interface code faster than a team can review it, and having them write the tests for it makes it challenging to know if the results are truly trustworthy or just the agent agreeing with itself.

Engineers and programs need independent verification that runs every time the code changes and leaves evidence a reviewer can use and trust.

YEARS OF ASSURED INTEGRATION, BUILT IN

Tenon packages the assured integration approach Tangram's engineers use on real defense programs into a tool suite your team runs itself. We've applied this technology across programs built on WOSA, AMS-GRA and OMS/UCI, on systems engineering models, and on both legacy and new codebases. Your engineers get the same method we've refined over years of delivering for our customers.

HOW IT WORKS



- 1. Define** the behavioral model in Flex, in your repo next to the code. Tenon checks it for errors and draws sequence and block diagrams from it, so engineers and program managers see the same picture.
- 2. Check** the design before code exists. Tenon model checks each protocol and returns Proven, Ill-formed, Indeterminate or Unsupported, pointing to the line and the reason when it finds an issue.
- 3. Test** the compiled software against the design, over the connection it really uses. No source code is needed, so it works on vendor deliveries.
- 4. Trace** every result back to your requirements. You can see which requirements have evidence behind them, which ones the evidence contradicts, and which nothing has checked yet, well before a design review asks.

Tenon saves this whole sequence as a workflow, one named recipe in your repo. Any engineer or AI agent can run it with a single command, and Tenon writes the GitLab pipeline that runs it on every commit, so the checks never depend on someone remembering the steps.

FROM INTEGRATION DESIGN TO EVIDENCE YOU CAN TRUST

 Faults caught in design Before code exists, with the line and the reason	 No source code needed Tests compiled software, including vendor deliveries	 Laptop and CI One saved workflow, a generated GitLab pipeline, no Docker-in-Docker	 Results where you expect them GitLab test reports, linked from every merge request	 Built for AI agents Built-in docs and machine-readable results an agent can act on
---	---	---	---	---

THE TENON ADVANTAGE

Approach	Limitation	Tenon Advantage
Hand-written tests	Slow to write, varies by engineer, usually tied to one message set	Tests generated from the spec, across standards and transports
Tests written by the AI that wrote the code	Results shift with the prompt and the model; a pass is hard to trust	An outside check with a spec-derived, repeatable strategy
Waiting for the lab or HWIL	Faults found at the most expensive point in the program	Design and code checked during development, on every commit
MBSE tools (Cameo, Rhapsody)	Describe the architecture, but don't test the code built to it	A behavioral model of the interfaces, checked and tested against the built software
Wiring point tools together	Your team owns the glue code, the version matching and the reporting	One install with a generated pipeline and a single report

TECH SPECS

PLATFORMS	macOS (Apple Silicon), Linux (x86_64), and Windows through WSL2
INSTALL	A single download with a checksum. Requires Docker
DESIGN CHECKING	Finds structural faults like deadlocks and out-of-order messages. Checks on message contents are on the roadmap
SOFTWARE TESTING	Tests your compiled software over TCP, UDP, ZeroMQ, Kafka, ActiveMQ, NATS or RabbitMQ. No source code needed
MESSAGE FORMATS	Binary, JSON, Protocol Buffers, XML (including OMS) and MAVLink, with program-specific formats on request
REQUIREMENTS TRACING	Shows which requirements are backed by evidence and which aren't
OUTPUTS	A browser dashboard, a PDF requirements verification matrix, diagrams, and test results GitLab and other CI tools can read. Reporting can't change a verdict
CI	Writes your GitLab pipeline for you. No Docker-in-Docker or privileged containers
AI AGENTS	Built-in documentation an LLM can read, plus machine-readable results and exit codes
LICENSING	Shared seats on your license, for people and CI jobs alike
SECURITY	Read-only inputs. A project can't ask for elevated privileges or host networking

Used by Tangram's own engineers and our customers on real defense programs. TRL 8 for interface verification in a software development and integration environment.

Want to see Tenon run against your own interfaces? Contact us at tangramflex.com.

Fault statistics: US Army DEVCOM Aviation & Missile Center, ACVIP: From Joint Multi-Role S&T to Future Vertical Lift (2021), slide 6.