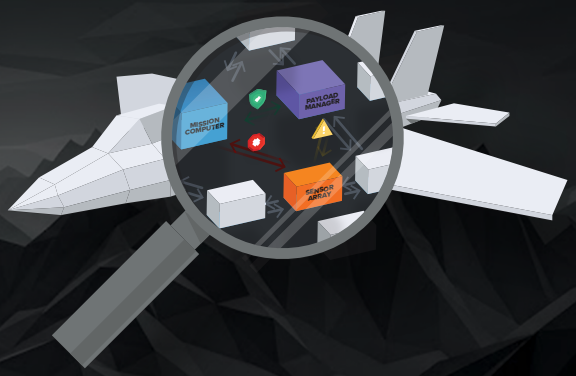
TANGRAMFLEX®

VERIFIER

CONTINUOUS SOFTWARE INTERFACE VERIFICATION



THE PROBLEM

90% of integration faults are introduced during design and development, but fewer than 20% are caught there. The rest surface at integration events, hardware-in-the-loop testing, or flight tests. That gets expensive, and burns weeks of scarce lab time.

Manual testing is slow, incomplete, and varies by engineer. AI-generated tests are non-deterministic, so you get different results from different prompts, different model versions, different days, etc. And not testing until the lab means gambling on hope.

Program teams need a way to continuously verify that the actual code implementation matches design, at the software level and during development, with results they can trust and defend. That's where the Tangram Pro™ Verifier comes into play.

PROVE YOUR SOFTWARE MATCHES THE DESIGN

Verifier automatically generates integration-level tests from Flex specs - formal, open-standard interface definitions that serve as the source of truth for your system. The same spec will generate the same tests every time - no prompt risk, no interpretation gaps, no variability between engineers or runs.

By continuously testing against Flex specs, Verifier offers rapid integration testing without compromising on assurance. It's not just showing the system works correctly, it's proving it.

HOW IT WORKS

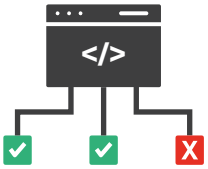
1. **Convert** static models, requirements, and ICDs into a formally verifiable Flex spec – establishing a single source of truth that defines how your component communicates with the world around it
2. **Auto-generate** repeatable verification tests directly from the spec – Verifier emulates the other side of every interface, playing the role of the systems your component expects to interact with
3. **Run** continuously to prove code conforms – locally or in CI pipelines – catching interface faults during development, not at expensive lab events

Verifier emulates the systems around your component, so you get **integration-level confidence** without waiting for the full system to be assembled.

Flex is an open, non-proprietary language for formally defining system interfaces. Machine-readable. Human-readable. Government-ownable.
Learn more at flexlang.org >>

RAPID, REPEATABLE, & ASSURED INTEGRATION TESTING

Same spec, same tests, every time
No engineer variability, no prompt risk



Catch faults during dev, not at the lab
Find and fix faults weeks earlier, 10x cheaper

Full traceability
Requirement → spec → test → result, auditable

Prove GRA compliance by behavior
Not checkbox conformance

Runs in your pipeline
CI/CD, CLI, and AI build-and-verify loops

THE VERIFIER ADVANTAGE

Approach	Limitation	Verifier Advantage
Manual integration testing	Slow, incomplete, varies by engineer	Automated, comprehensive, repeatable
AI-generated tests	Non-deterministic, prompt-dependent, hard to audit	Same spec → same quality tests every time, full traceability
Testing only at lab/HWIL	Faults caught too late, 10x cost to fix	Continuous verification during development
Generic test frameworks	Not GRA-aware, no formal spec basis	Defense-native, formal spec-driven, GRA compliance built in
MSBE tools (Cameo, Rhapsody)	Model-level only, doesn't test code	Bridges model to code - tests actual implementation

TECH SPECS

INPUTS	Flex specifications (formal DSL), ICDs, MBSE models, GRA standard definitions
OUTPUTS	Executable integration test suites, traceability artifacts (verification reports, compliance evidence)
DELIVERY	Cloud-hosted UI, CI/CD pipeline integration, CLI
VERIFICATION SUPPORT	Repeatable test generation from formal specifications - not probabilistic, not AI-based
GRA SUPPORT	WOSA, OMS/UCI, AMS-GRA, FACE, and many others
CLASSIFICATION	Unclassified through TS/SCI, including IL5/6 air-gapped deployments
AI INTEGRATION	Verifies AI-generated code conforms to spec (assurance); provides consistent, repeatable feedback for build-and-verify loops
TRACEABILITY	Requirement → Flex spec → generated test → test result (full chain, auditable)

Verifier has been proven on programs across weapons, EW, platform integration, and autonomy. It's built for defense integration: GRA-aware, operational in IL5/6 classified environments, and used daily by Tangram's own 70-person engineering team.