Automated Evidence Generation for Continuous Certification

High Confidence Software and Systems 2022

Virtual, May 18th 2022

Mauricio Castillo-Effen, Ph.D.

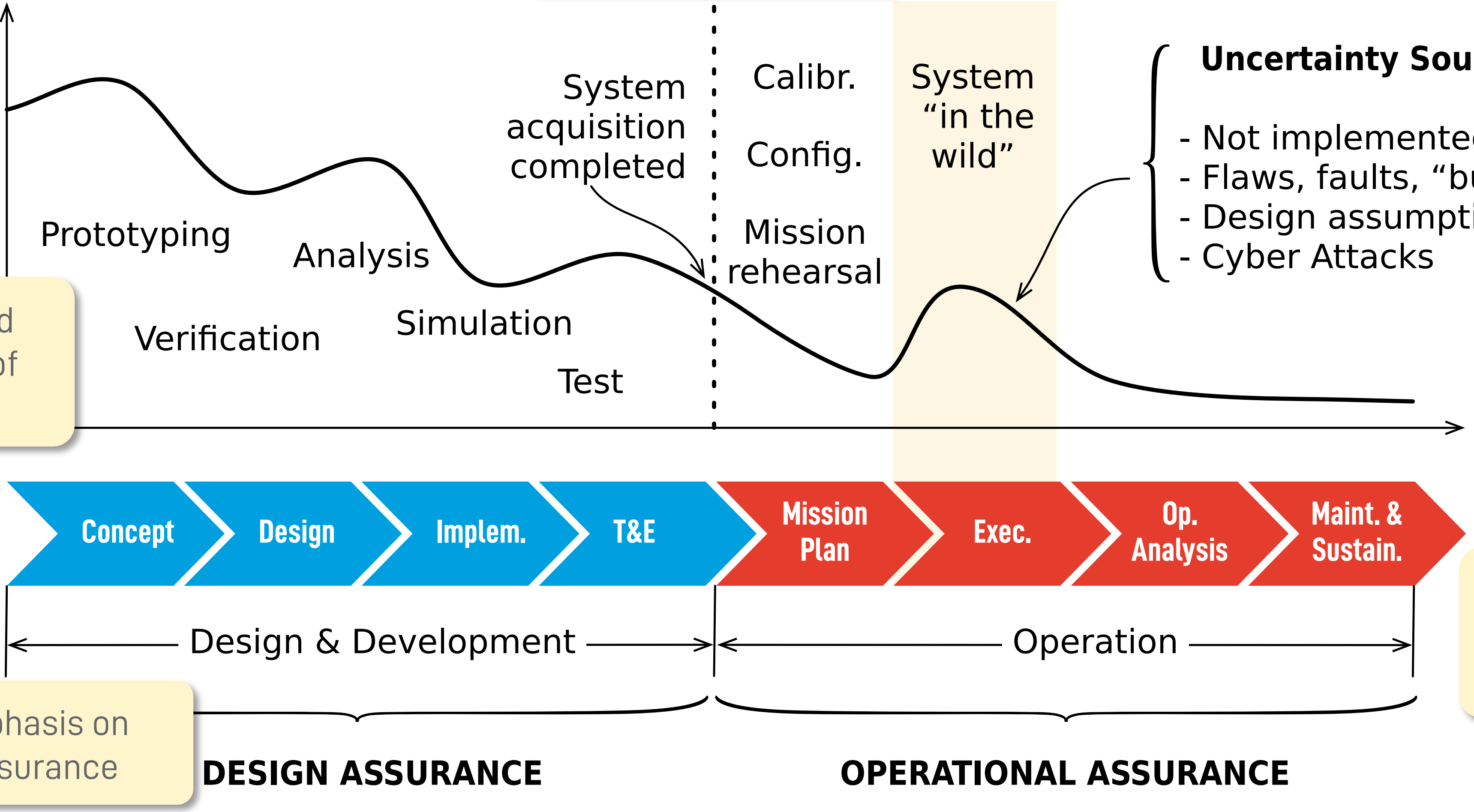
LOCKHEED MARTIN

LOCKHEED MARTIN PUBLIC INFORMATION

Overview

1. Continuous Assurance
2. CertGATE: An Evidence Generation Workbench
3. CertGATE Technology
4. Example

Assurance Today



Deployed capability addressing "all foreseeable conditions"

Uncertainty Sources

- Not implemented features
- Flaws, faults, "bugs"
- Design assumption violations
- Cyber Attacks

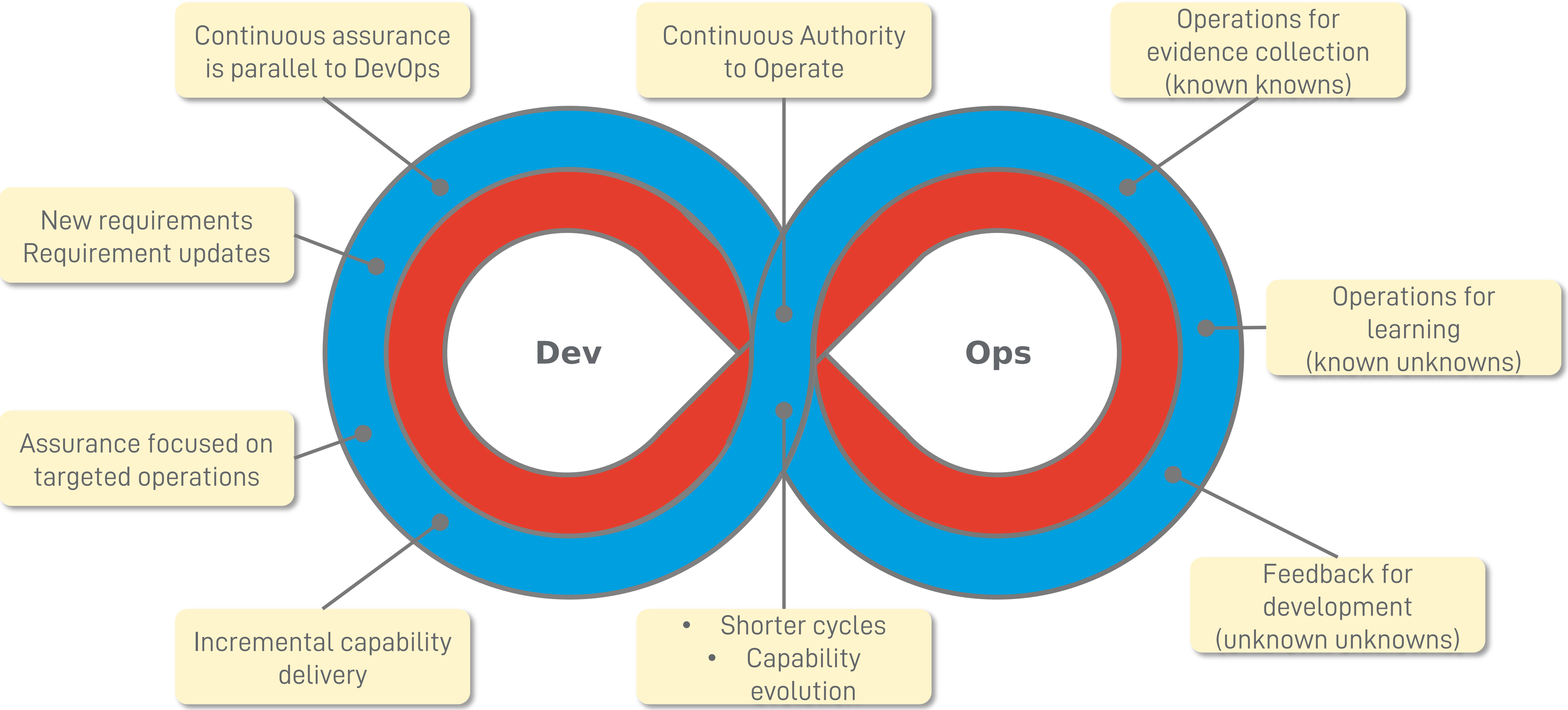
Rigid / predefined accepted forms of evidence

Long cycles

Feedback does not go back to "Design & Development"

Large emphasis on design assurance

Assurance Tomorrow



Continuous assurance is parallel to DevOps

Continuous Authority to Operate

Operations for evidence collection (known knowns)

New requirements
Requirement updates

Operations for learning (known unknowns)

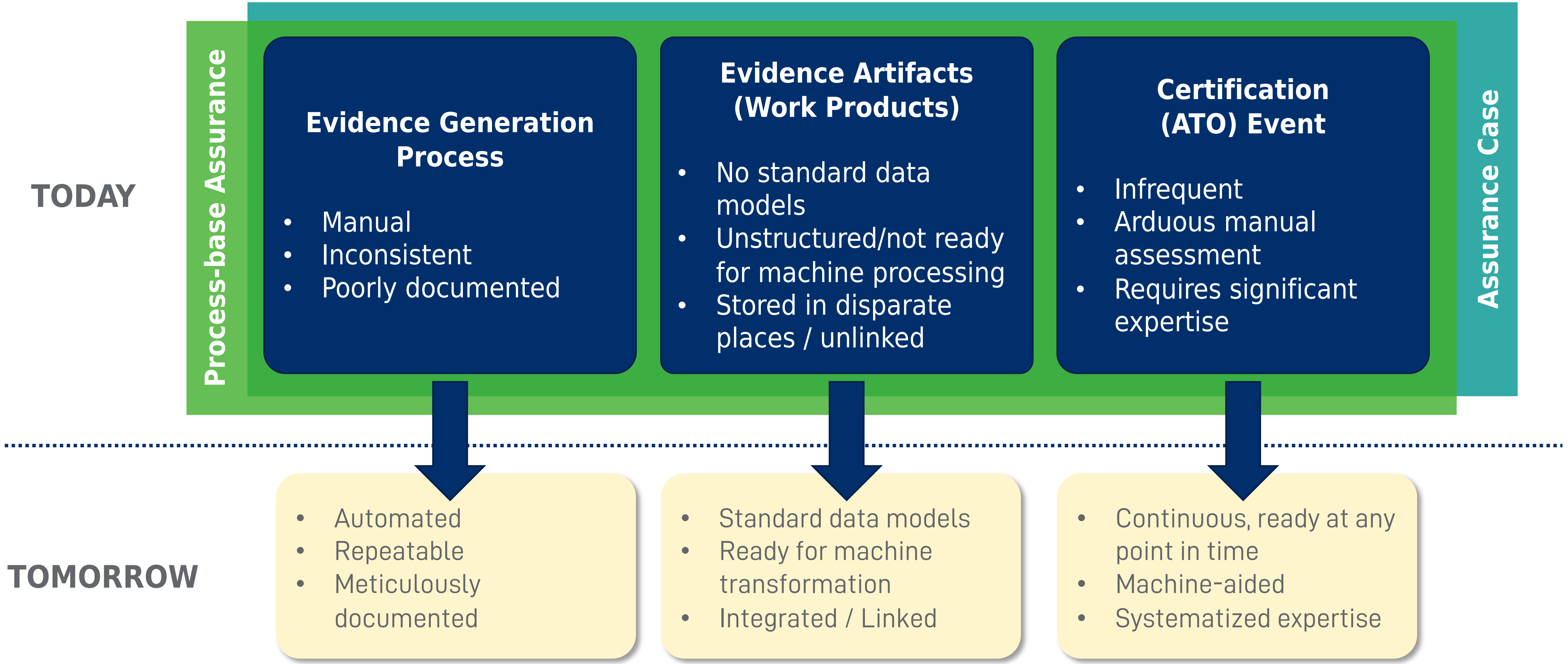
Assurance focused on targeted operations

Feedback for development (unknown unknowns)

Incremental capability delivery

- Shorter cycles
- Capability evolution

Design Assurance



Process-base Assurance

Assurance Case

TODAY

Evidence Generation Process

- Manual
- Inconsistent
- Poorly documented

Evidence Artifacts (Work Products)

- No standard data models
- Unstructured/not ready for machine processing
- Stored in disparate places / unlinked

Certification (ATO) Event

- Infrequent
- Arduous manual assessment
- Requires significant expertise

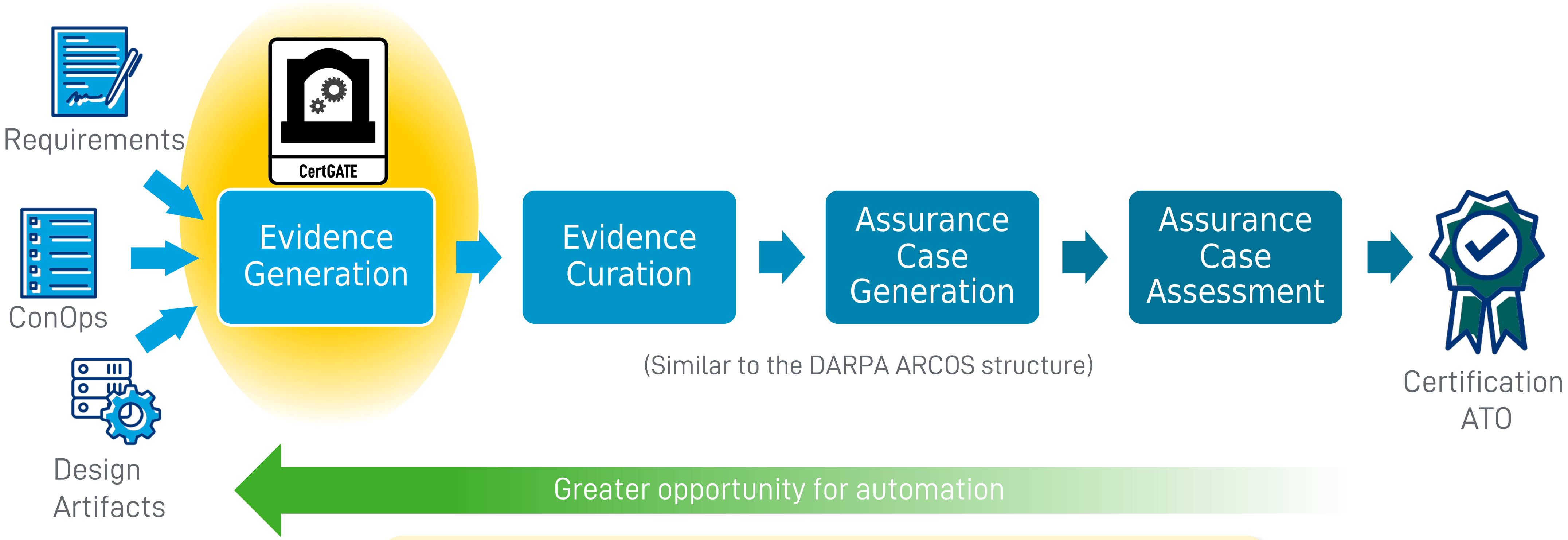
TOMORROW

- Automated
- Repeatable
- Meticulously documented

- Standard data models
- Ready for machine transformation
- Integrated / Linked

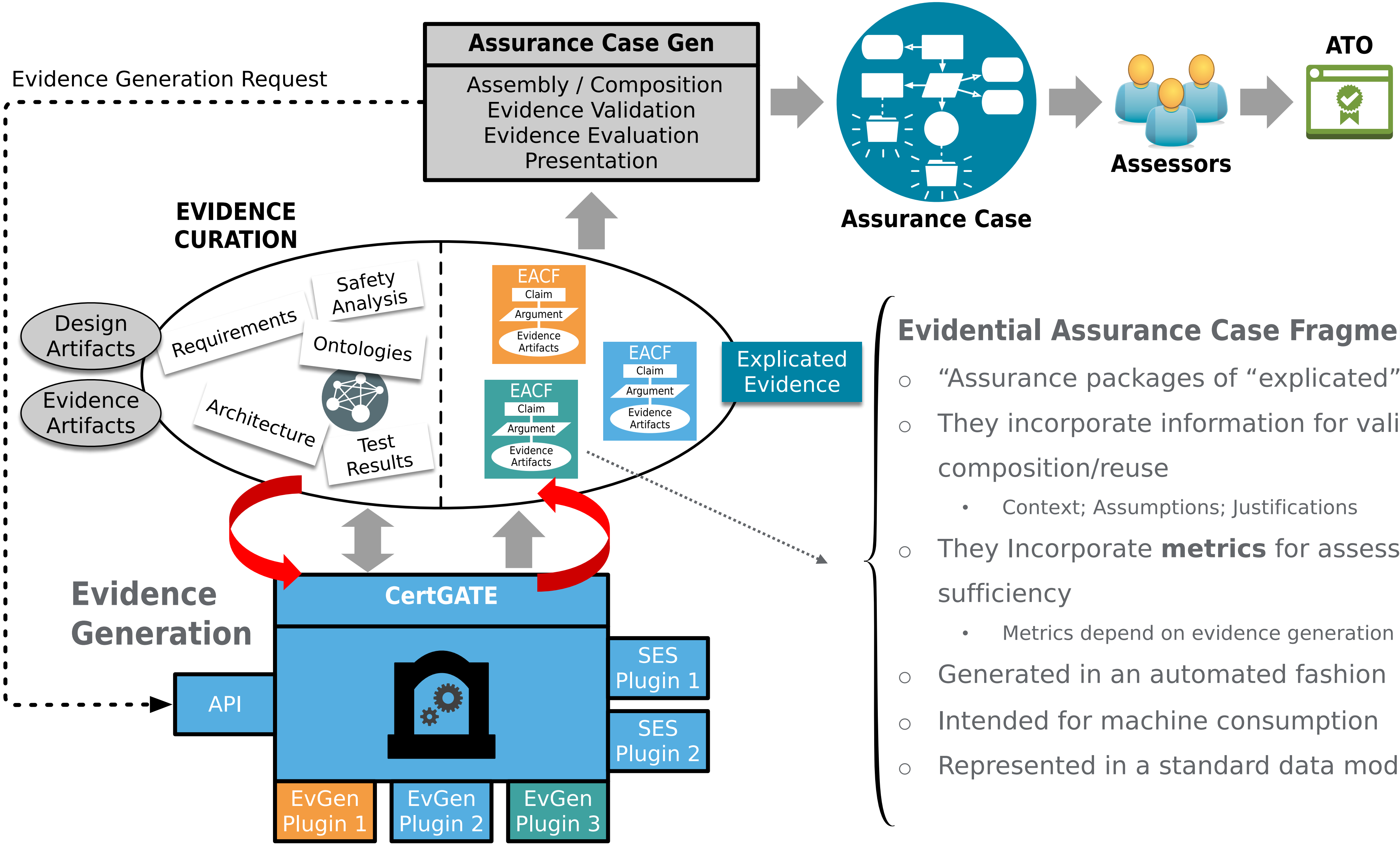
- Continuous, ready at any point in time
- Machine-aided
- Systematized expertise

Technology Areas in Design Assurance


Requirements
ConOps
Design Artifacts
CertGATE
Evidence Generation
Evidence Curation
Assurance Case Generation
Assurance Case Assessment
Certification ATO
(Similar to the DARPA ARCOS structure)
Greater opportunity for automation

Two Pillars to CertGATE Automation:

1. Evidential Assurance Case Fragments (EACFs)
2. Evidence Generation Language



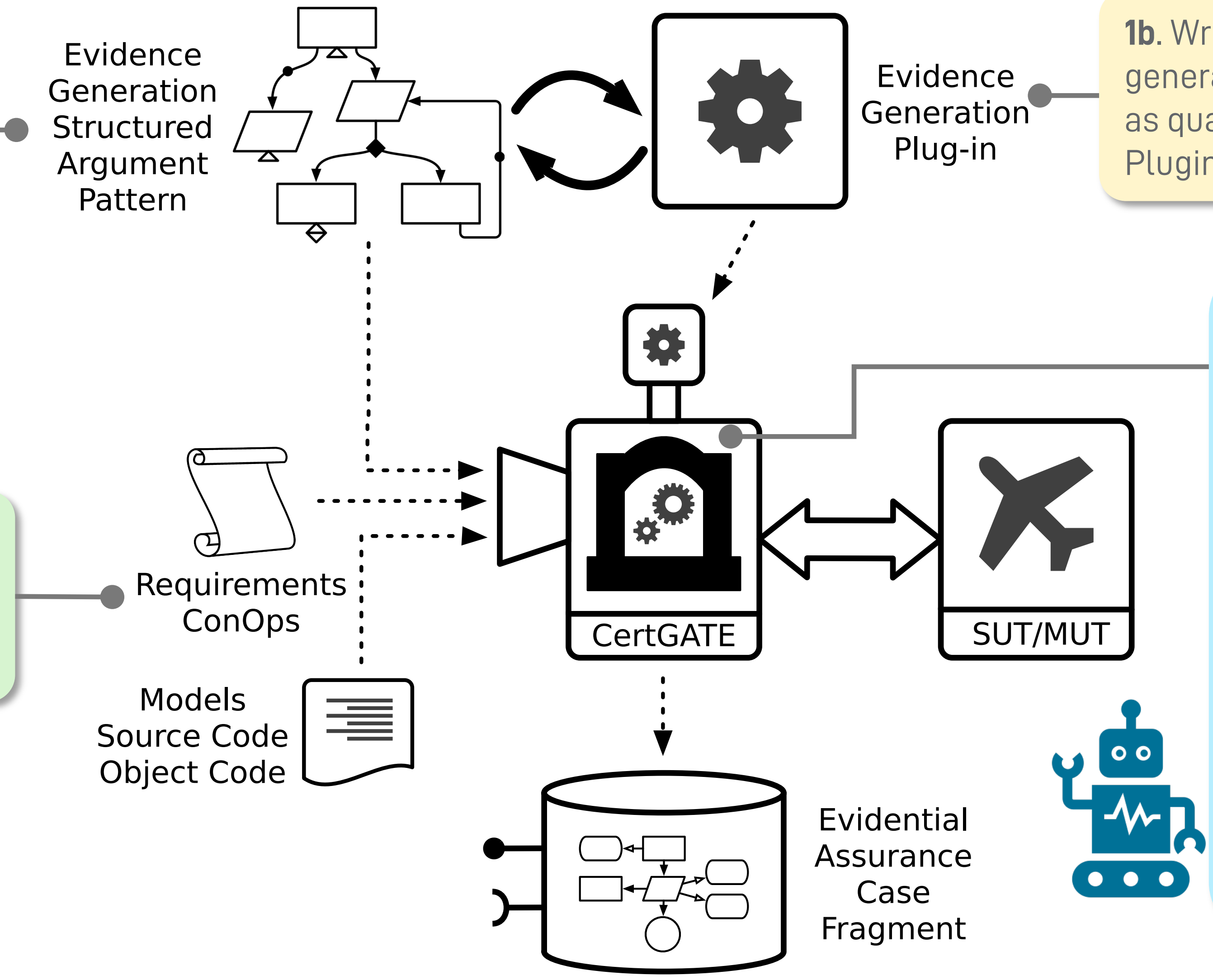
Evidential Assurance Case Fragments (EACFs)

- “Assurance packages of “explicated” evidence
- They incorporate information for valid composition/reuse
  - Context; Assumptions; Justifications
- They Incorporate **metrics** for assessment of sufficiency
  - Metrics depend on evidence generation technique
- Generated in an automated fashion
- Intended for machine consumption
- Represented in a standard data model (SACM)

CertGATE Workflow

1a. Encoding of evidence generation process as a structured argument pattern

1b. Wrap evidence generation technique as qualified CertGATE Plugin


Evidence Generation Structured Argument Pattern
Evidence Generation Plug-in
Requirements ConOps
Models Source Code Object Code
CertGATE
SUT/MUT
Evidential Assurance Case Fragment

2. Definition of workflows and input artifacts

3. Transformation of patterns to argument instances
4. Transformation of argument instance to a set of Evidence Generation directives in EGL
5. Execution of EGL directives
6. Connect evidence artifacts to structured argument
7. Package EACF

Evidence Generation Language (EGL)

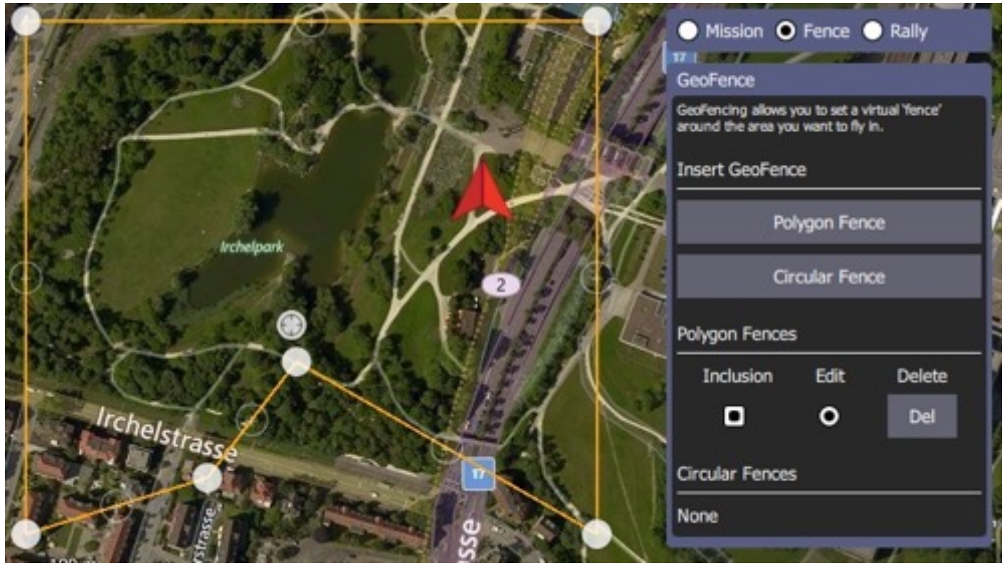
- Internal Domain-specific Language (DSL).
- Prototype written in Python.
- Defines, parameterizes, evidence generation actions, inputs, and outputs.

What it is

- Operations supported by CertGATE plugins
- Test generation; test execution; code instrumentation; software-defined SES configuration; parameterized static analysis; test coverage computation; transformation operations; interactions with code repos, etc.

Sample Operations

Evidence Generation Workflow Example



○ Mission ● Fence ○ Rally

GeoFence

GeoFencing allows you to set a virtual 'fence' around the area you want to fly in.

Insert GeoFence

Polygon Fences

Inclusion	Edit	Delete
■	●	Del

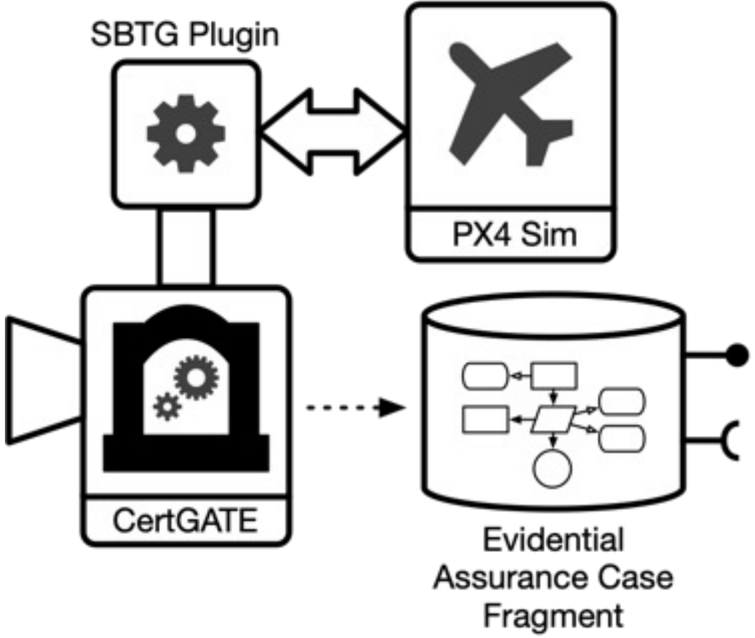
Circular Fences

None

We want to produce evidence that demonstrates that the PX4's GNC module never violates geofence constraints. We will vary the geofence geometry (G), the takeoff and landing locations (P_S, P_L), the UAS speed (V_{UAS}) and wind, modeled as having a direction and a speed (W_D, W_S).
Plugin: Search-based Test Generation (SBTG)

Inputs

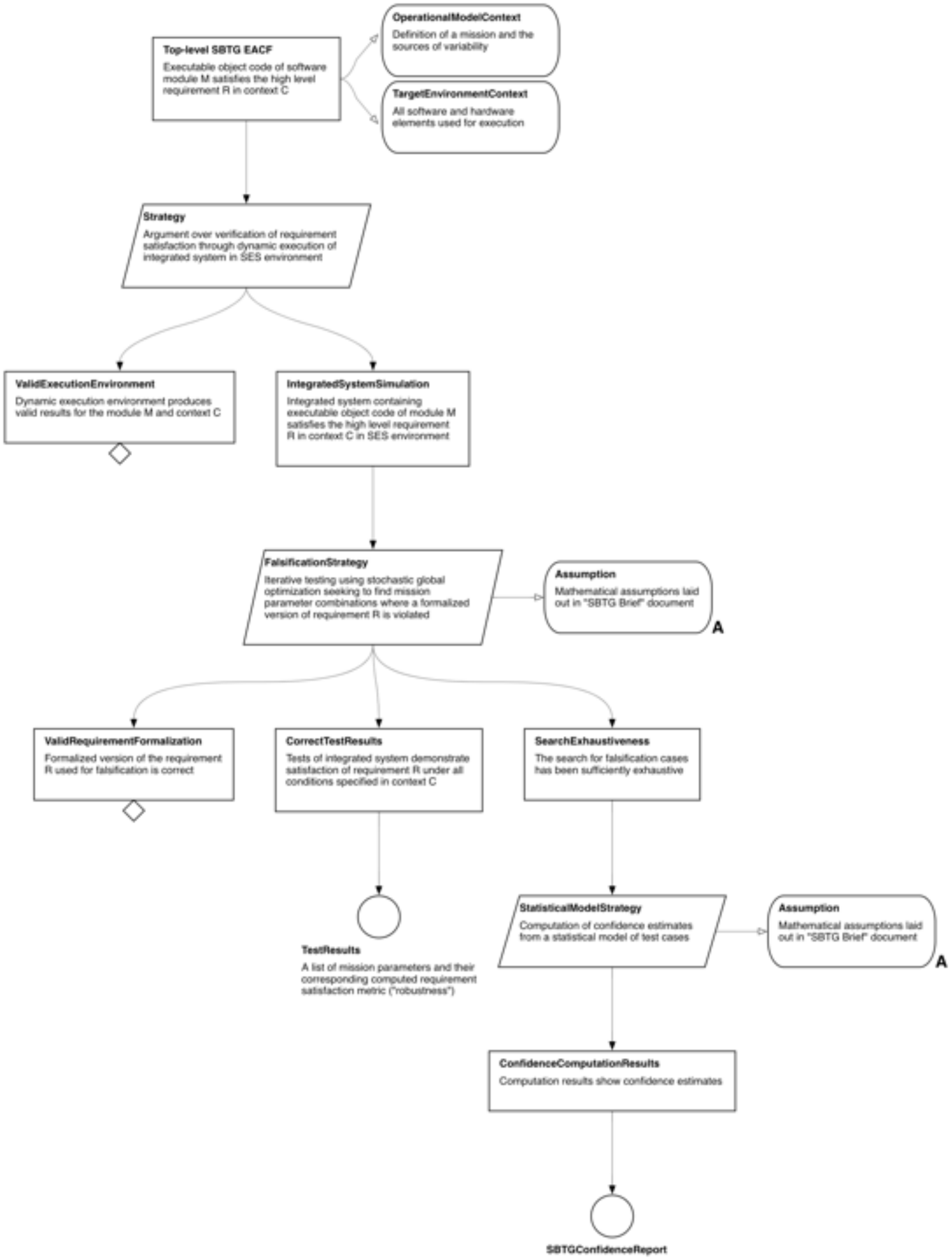
S:	PX4_GNC
C:	*Operational Model*: PX4_Scenario *Target Environment*: HighFidelityPX4
R:	The system never violates the GeoFence constraints $\square(P \in G)$
M:	SBTG_STALIRO



Evidence Artifacts

- Tests: Scenario parameters (G, P_S, P_L, V_{UAS} , W_D, W_S), requirement satisfaction metric ("robustness" ρ)
- SBTG Confidence report

SBTG Fragment Pattern (Natural Language)



Top-level SBTG EACF
Executable object code of software module M satisfies the high level requirement R in context C

OperationalModelContext
Definition of a mission and the sources of variability

TargetEnvironmentContext
All software and hardware elements used for execution

Strategy
Argument over verification of requirement satisfaction through dynamic execution of integrated system in SES environment

ValidExecutionEnvironment
Dynamic execution environment produces valid results for the module M and context C

IntegratedSystemSimulation
Integrated system containing executable object code of module M satisfies the high level requirement R in context C in SES environment

FalsificationStrategy
Iterative testing using stochastic global optimization seeking to find mission parameter combinations where a formalized version of requirement R is violated

Assumption
Mathematical assumptions laid out in "SBTG Brief" document

ValidRequirementFormalization
Formalized version of the requirement R used for falsification is correct

CorrectTestResults
Tests of integrated system demonstrate satisfaction of requirement R under all conditions specified in context C

SearchExhaustiveness
The search for falsification cases has been sufficiently exhaustive

TestResults
A list of mission parameters and their corresponding computed requirement satisfaction metric ("robustness")

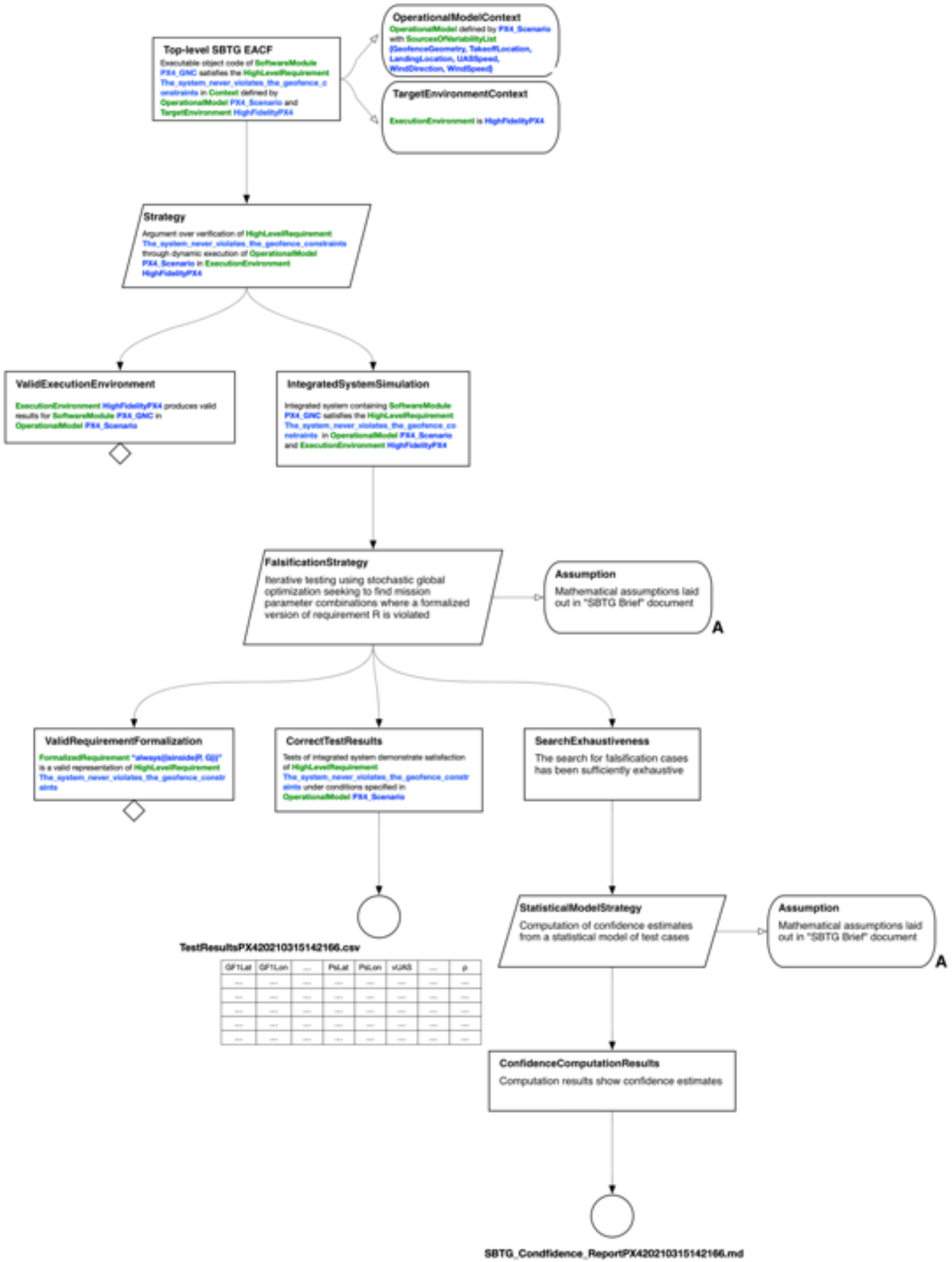
StatisticalModelStrategy
Computation of confidence estimates from a statistical model of test cases

Assumption
Mathematical assumptions laid out in "SBTG Brief" document

ConfidenceComputationResults
Computation results show confidence estimates

SBTGConfidenceReport

Instantiated SBTG Fragment Pattern (Natural Language)



Top-level SBTG EACF
Executable object code of SoftwareModule PX4_GNC satisfies the HighLevelRequirement The_system_never_violates_the_geofence_constraints in Context defined by OperationalModel PX4_Scenario and TargetEnvironment HighFidelityPX4

OperationalModelContext
OperationalModel defined by PX4_Scenario with SourcesOfVariabilityList [GeofenceGeometry, TakeoffLocation, LandingLocation, UASSpeed, WindDirection, WindSpeed]

TargetEnvironmentContext
ExecutionEnvironment is HighFidelityPX4

Strategy
Argument over verification of HighLevelRequirement The_system_never_violates_the_geofence_constraints through dynamic execution of OperationalModel PX4_Scenario in ExecutionEnvironment HighFidelityPX4

ValidExecutionEnvironment
ExecutionEnvironment HighFidelityPX4 produces valid results for SoftwareModule PX4_GNC in OperationalModel PX4_Scenario

IntegratedSystemSimulation
Integrated system containing SoftwareModule PX4_GNC satisfies the HighLevelRequirement The_system_never_violates_the_geofence_constraints in OperationalModel PX4_Scenario and ExecutionEnvironment HighFidelityPX4

FalsificationStrategy
Iterative testing using stochastic global optimization seeking to find mission parameter combinations where a formalized version of requirement R is violated

Assumption
Mathematical assumptions laid out in "SBTG Brief" document
A

ValidRequirementFormalization
FormalizedRequirement "always(!inside(P, G))" is a valid representation of HighLevelRequirement The_system_never_violates_the_geofence_constraints

CorrectTestResults
Tests of integrated system demonstrate satisfaction of HighLevelRequirement The_system_never_violates_the_geofence_constraints under conditions specified in OperationalModel PX4_Scenario

SearchExhaustiveness
The search for falsification cases has been sufficiently exhaustive

TestResultsPX420210315142166.csv

GF1Lat	GF1Lon	...	PsLat	PsLon	vUAS	...	ρ
—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—

StatisticalModelStrategy
Computation of confidence estimates from a statistical model of test cases

Assumption
Mathematical assumptions laid out in "SBTG Brief" document
A

ConfidenceComputationResults
Computation results show confidence estimates

SBTG_Condfidence_ReportPX420210315142166.md

Summary

Continuous Assurance

- It is NOT just “old assurance + CI/CD pipelines”
- We need to rethink the size of what is delivered
- Operations as additional source of learning and evidence

Evidential Assurance Case Fragments

- Assurance packages of explicated evidence
- Separate concerns between automated activities (e.g.: generating evidence) and interactive activities (e.g.: constructing/assessing assurance cases)
- Apply reuse and composition

“Digital” Assurance Cases

- Are built from EACFs
- They capture a snapshot of the system’s assurance state
- Could be used to generate assessable assurance cases (e.g.: GSN, CAE, FAN, etc.) or other views useful for evaluating benefit-risk ratios

CertGATE

- Evidence Generation workbench that generates EACFs
- Extensible thanks to plugin architecture and Evidence Generation Language
- An enabler for continuous assurance
- Uses argument patterns as evidence generation “recipes”

LOCKHEED MARTIN