Distributed MILS (D-MILS) Technology Achievements

Rance DeLong

THE Open GROUP
Making standards work®

A Brief MILS/D-MILS Refresher

- MILS is a component-based approach for the construction, assurance, and certification of dependable systems that encourages a commercial marketplace of off-the-shelf high-assurance components
- MILS can be understood as a two phase approach:
  - Architecture
    - Abstract policy architecture represented with "boxes" (operational components) and "arrows" (interactions)
    - System purpose is achieved by behavior of the operational components and their interactions
    - Assumption: the architecture will be strictly enforced
  - Implementation
    - A robust resource-sharing platform composed of MILS foundational components creates strongly isolated "exported resources"
    - Components individually developed and assured according to standard specifications
    - Components compose "additively" to form a distributed trusted sharing substrate, the MILS Platform

MILS Policy Architecture

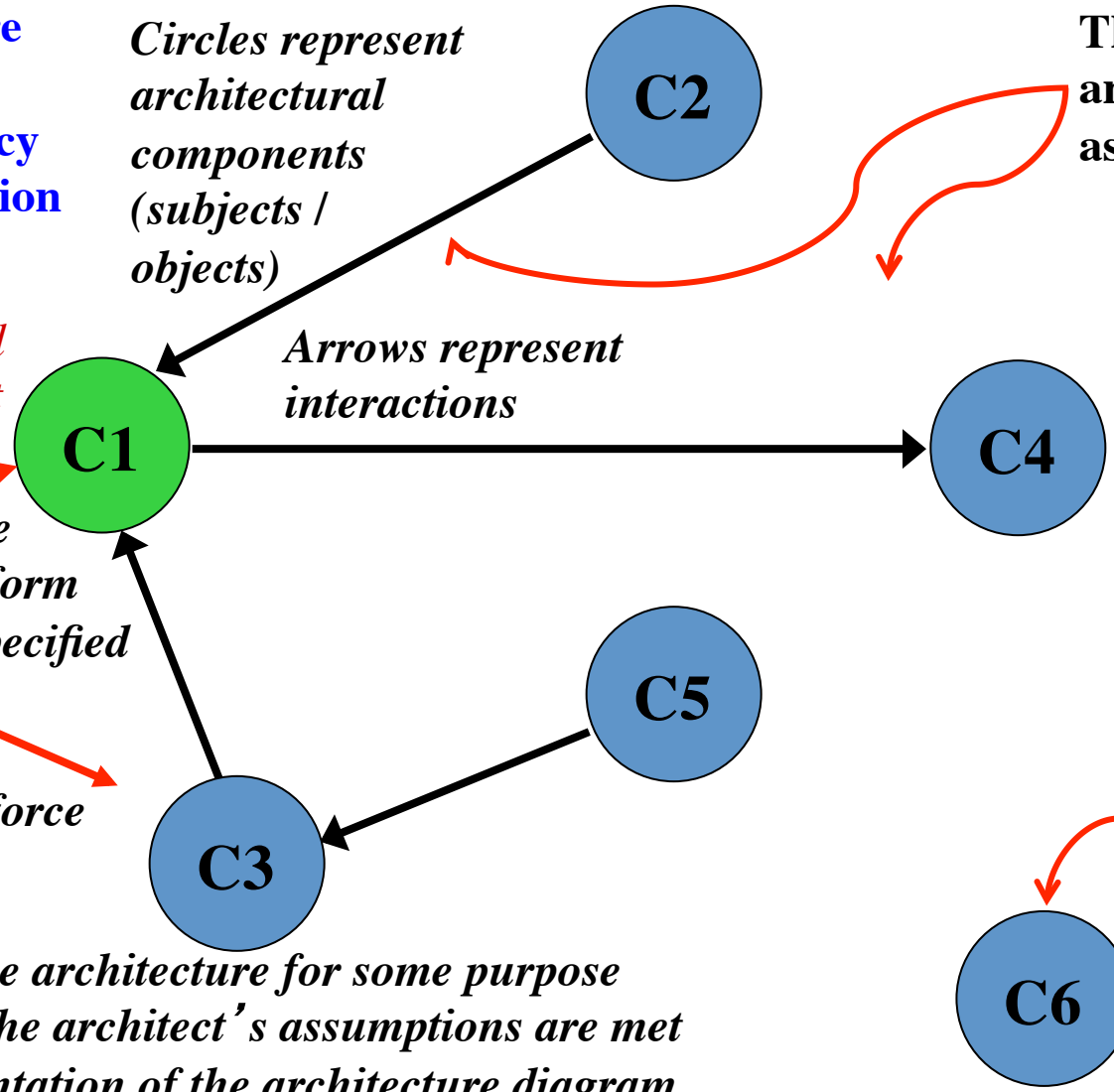
The architecture expresses an interaction policy among a collection of components

Circles represent architectural components (subjects / objects)

The *absence* of an arrow *is as significant* as the *presence* of one

Trusted Subject

Arrows represent interactions

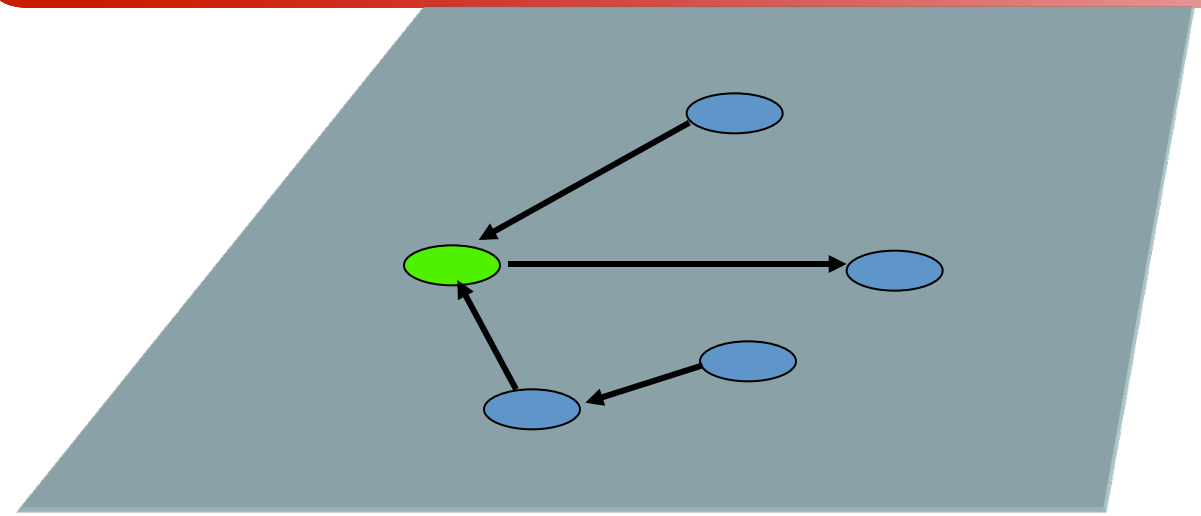


Components are assumed to perform the functions specified by the architect (trusted components enforce a local policy)

This component has *no interaction* with any other

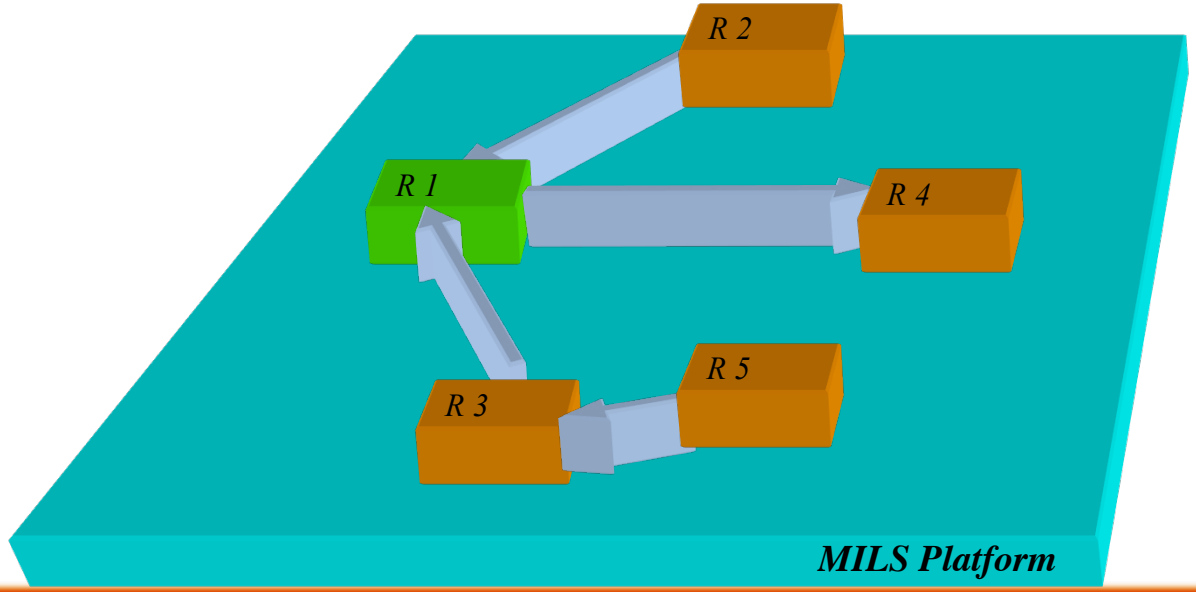
Suitability of the architecture for some purpose presumes that the architect's assumptions are met in the implementation of the architecture diagram.

MILS Platform – Provides Straightforward Implementation of Policy Architecture



Architecture

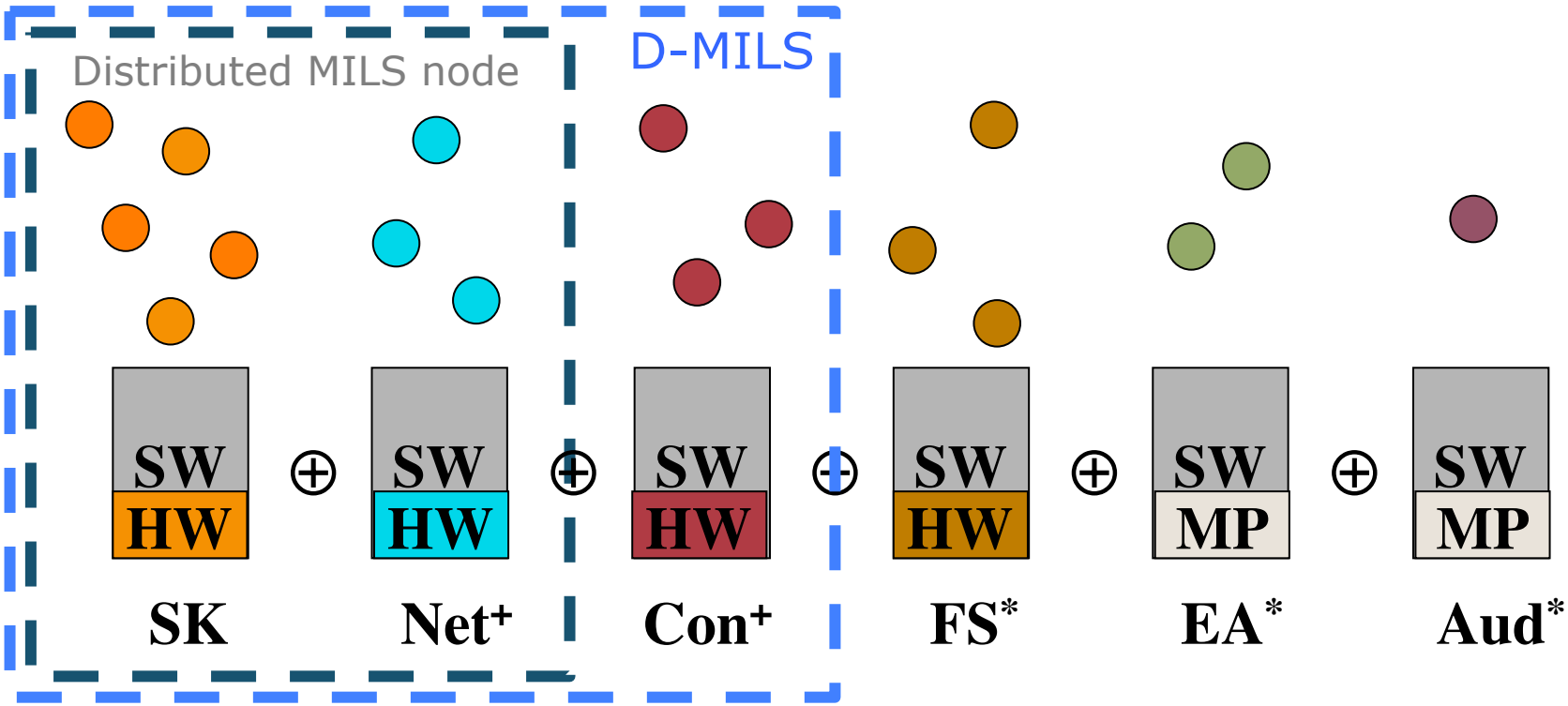
Validity of the architecture assumes that the *only* interactions of the circles (operational components) is through the arrows depicted in the diagram

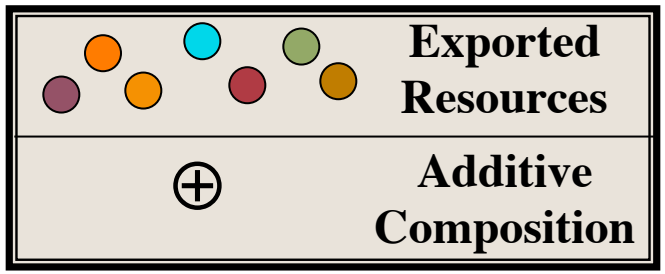


Implementation

SK, with other MILS foundational components, form the *MILS Platform* allowing operational components to share physical resources while enforcing Isolation and Information Flow Control

The MILS Platform: Resource-Sharing Components





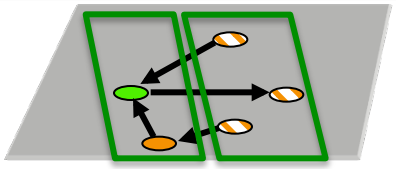
additive compositionality – e.g.,
Partitioning Kernel ⊕ Partitioning Net
= Partitioning (Kernel + Net)

MP = MILS Platform

+ D-MILS includes limited versions of Net(work) and Con(sole)

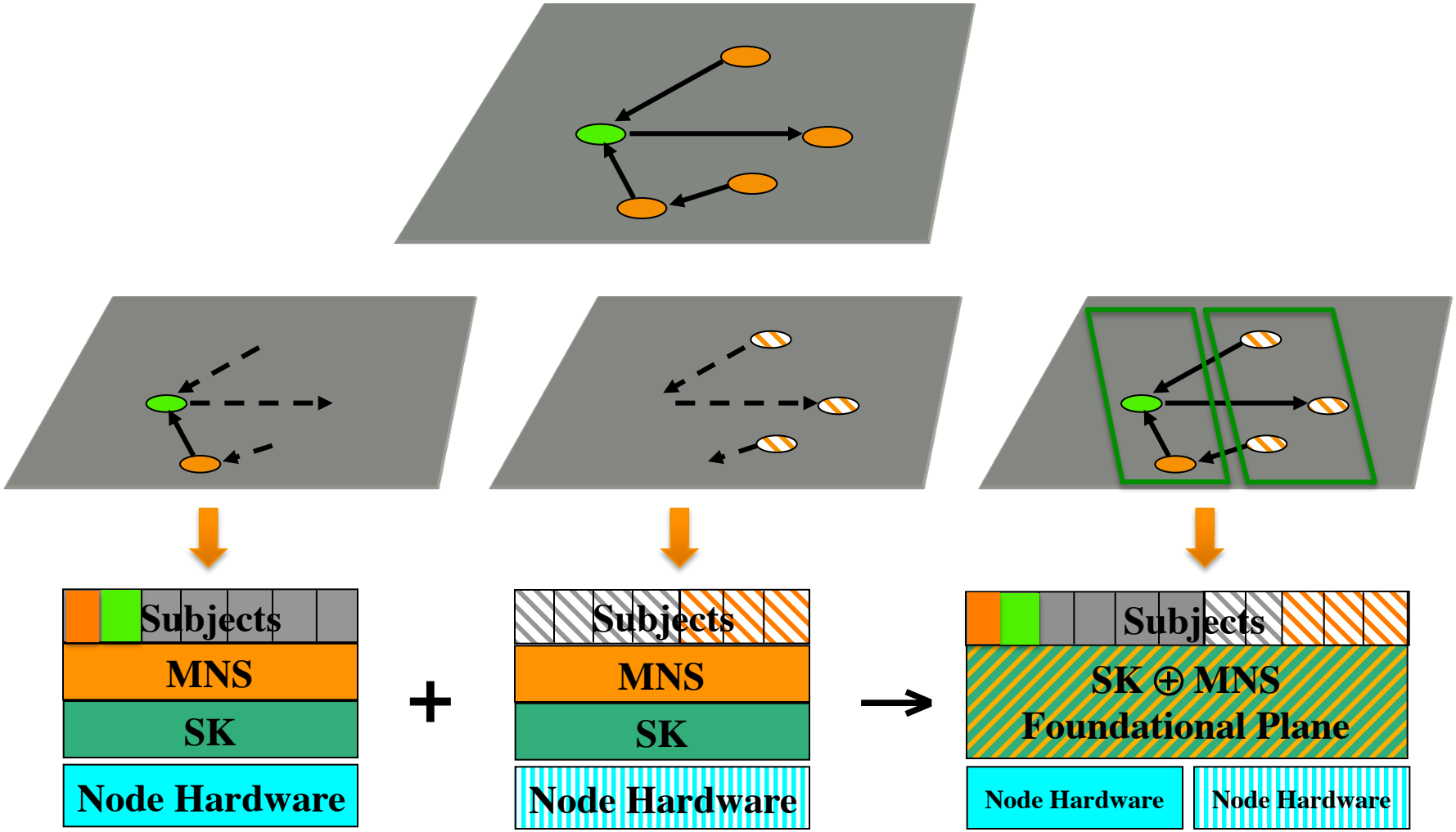
* D-MILS does not include MILS FS, EA and Aud components

Distributed MILS



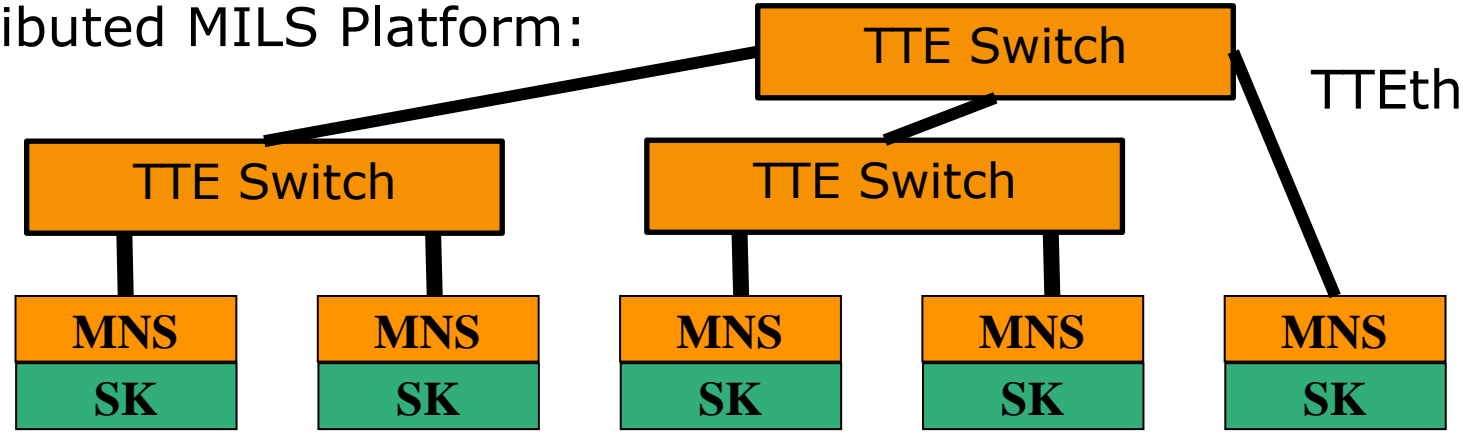
- A single policy architecture may span multiple D-MILS nodes expressed in declarative MILS-AADL
- Guarantees similar to a single MILS node: isolation, information flow control, determinism
- Determinism over network could be achieved in various ways – in D-MILS we use Time-Triggered Ethernet (TTE)
- Must configure and schedule the network and the processors of the nodes coherently
- Support verification of architectural properties, presentation of assurance case, and generation of configuration with integrated automation with the greatest practical use of existing verification technology

Distributed MILS:
Policy architecture deployment spanning nodes

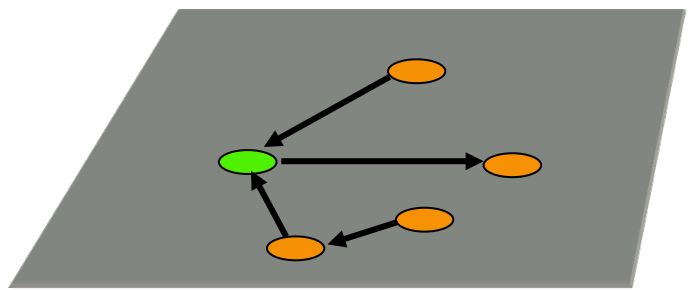


Distributed MILS Platform – MILS nodes with deterministic communication

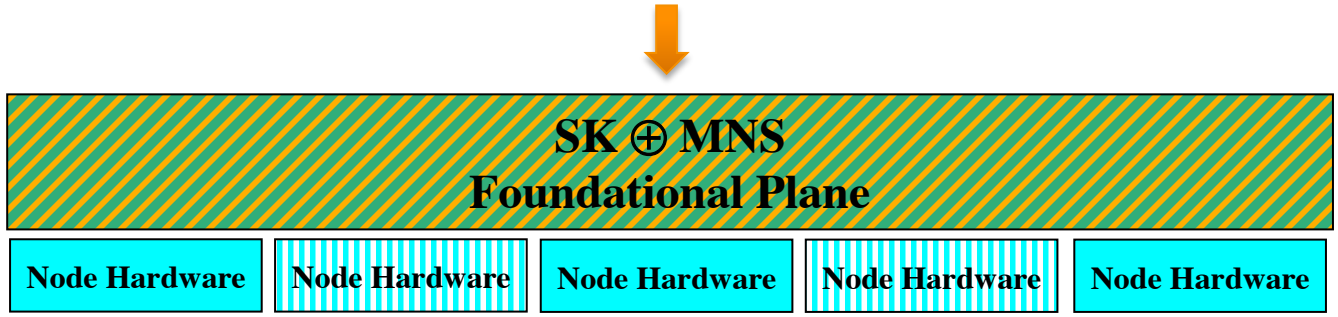
A Distributed MILS Platform:



Enables:



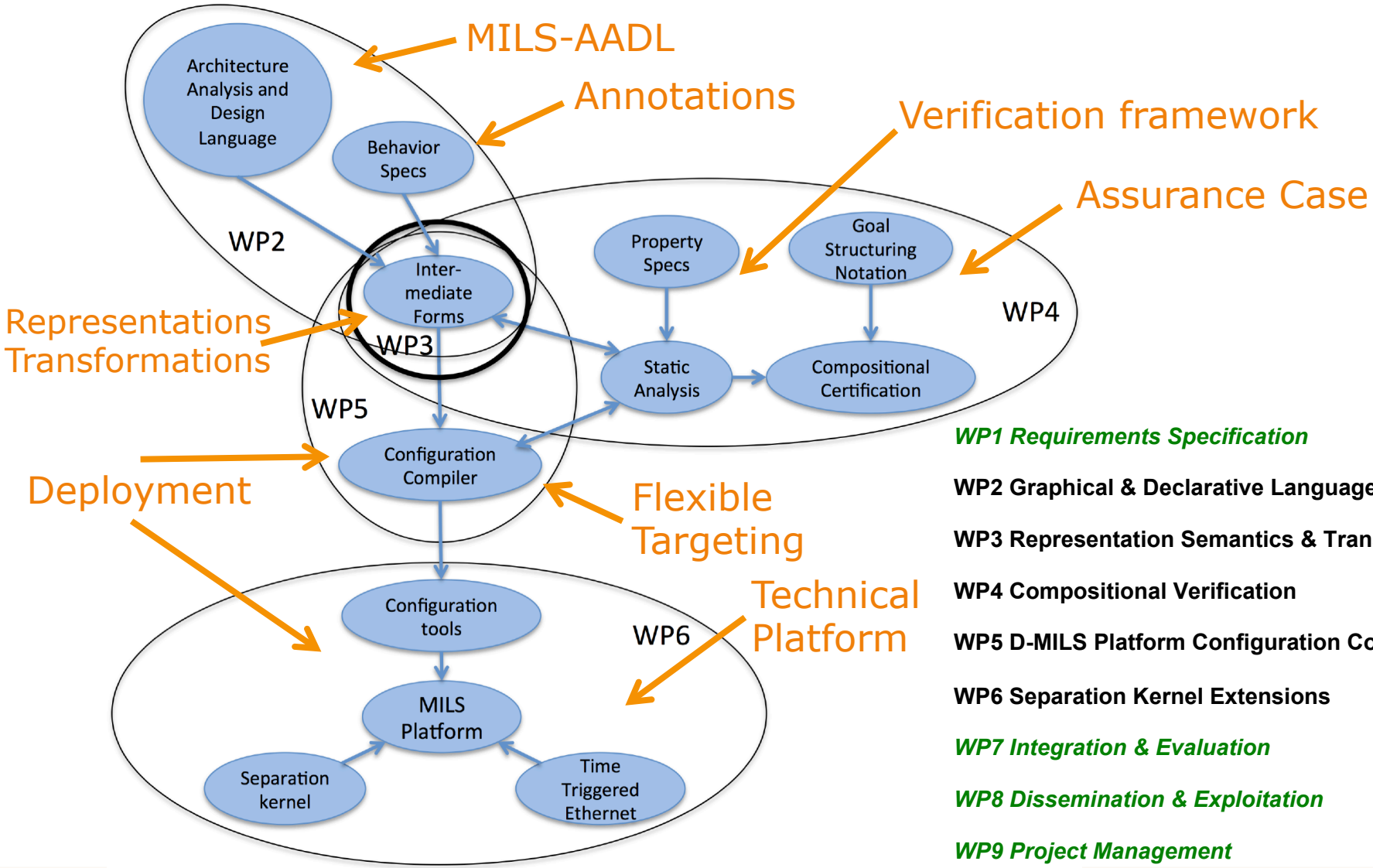
Realization of deterministic distributed MILS architectures



D-MILS Technology Areas

- Graphical and Declarative Languages (front-end)
  - Architecture Analysis and Design Language, MILS extended subset (MILS-AADL)
  - Goal Structuring Notation (GSN)
- Integration of GSN and MILS-AADL
  - Structure of GSN assurance case informed by information gleaned from MILS-AADL model
- Representations and Transformations
  - Semantics preserving transformation between front-end languages and intermediate and back-end languages of the analysis tools
- Compositional Verification
  - Reduce verification of system to independent verification of its parts
  - Properties to verify and appropriate verification strategies and tools
- Compositional Assurance Cases
  - Modular GSN
  - Rely / guarantee argumentation
- Configuration Compiler
  - Generate configuration information for D-MILS nodes and connecting TT network infrastructure
  - Configuration constrained by actual physical resources available
- D-MILS Platform
  - Separation kernel with extended configuration
  - MILS Network Subsystem with TTEthernet

Technology Work Package Relationships



WP1 Requirements Specification

WP2 Graphical & Declarative Languages

WP3 Representation Semantics & Transforms

WP4 Compositional Verification

WP5 D-MILS Platform Configuration Compiler

WP6 Separation Kernel Extensions

WP7 Integration & Evaluation

WP8 Dissemination & Exploitation

WP9 Project Management

Summary of D-MILS Project Technology Achievements

- Coordinated "top-to-bottom" and "end-to-end" strategy has been demonstrated – from declarative system description language, through verification, to platform deployment and execution
- Basis for increased confidence that system meets requirements, through modelling, analysis, and construction.
- D-MILS Declarative language
  - MILS-AADL extended subset of standard AADL, syntax and formal semantics
  - compiler implementation with translations to languages of the backend tools in the D-MILS tool chain
- D-MILS Assurance Case
  - Automated support for compositional assurance with modular assurance case guided by declarative language features
  - Integration of verification results
- Compositional Verification
  - Application of Contract-Based Design (CBD) to MILS – critical to MILS refinement claims
  - Integrated framework providing a high degree of verification automation and simulation
- D-MILS Configuration Compiler
  - Automated configuration of distributed platform to distributed mode
  - Coherent details of system deployment across multiple nodes
  - Method to ensure configuration correctness
- D-MILS Distributed Platform
  - Operational deterministic distributed platform based on:
  - Extended separation kernel (LynxSecure)
  - MILS Network System
  - Extended deterministic networking (TTEthernet)
  - MILS Console System
- D-MILS Industrial Demonstration and Evaluation
  - Use and assessment of D-MILS tool chain for modeling/verification/deployment of two industrial use cases
  - fortiss Smart Microgrid
  - Frequentis Voice Services

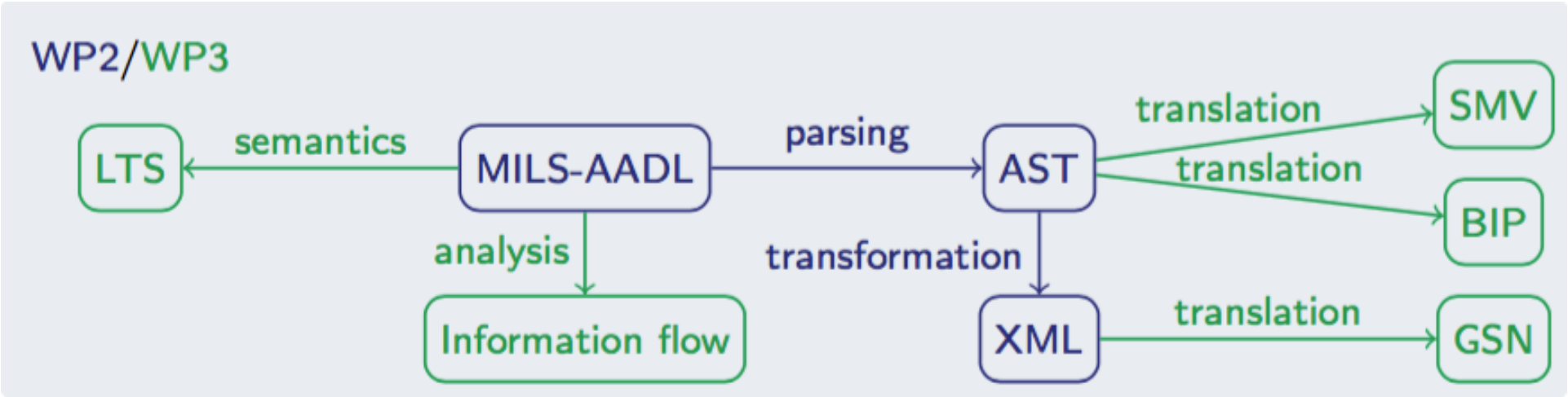
More about selected accomplishments

WP2 Declarative Language & WP3 Semantics and Transformations

- Prior to D-MILS
  - SLIM dialect of AADL
  - COMPASS analysis tools
- Accomplishments of D-MILS
  - MILS-AADL syntax and formal semantics including MILS features, ops and data types
  - Translations to SMV/BIP/Prolog
  - Integration of annotations for property specifications, contracts, and deployment
  - Analysis of cryptographically masked info flow
  - Slicing-based verification procedure for info flow properties supporting encryption

WP2 Declarative Language & WP3 Semantics and Transformations

- WP2: Languages
  - Definition and parsing of MILS-AADL
  - Extended data types and operations
  - XML metamodel and export

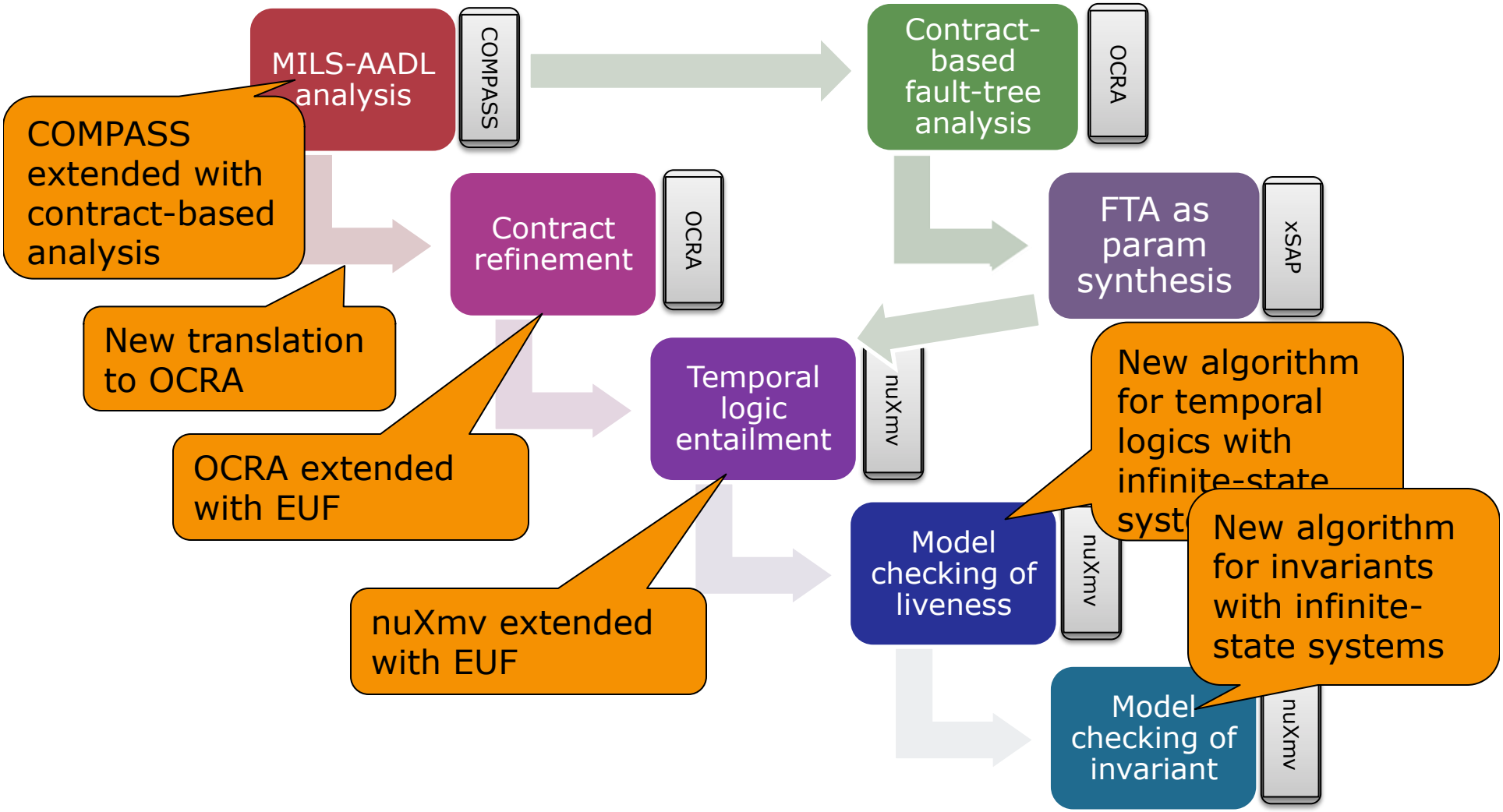


- WP3: Semantics and Transformations
  - Definition of formal semantics
  - Support for translation to SMV/BIP/Prolog and XML for GSN

WP4 Compositional Verification

- Prior to D-MILS, these tools and their analyses
  - COMPASS
  - nuSMV, OCRA
  - BIP
- Accomplishments of D-MILS
  - Integrated verification framework
  - Contract-Based Design applied to MILS-AADL
  - New algorithms, including invariants and temporal logics
  - New and extended properties and compositional analyses
    - Functional requirements
    - Security
    - Safety
    - Real-time
    - Hybrid systems
    - Timed systems
    - Uniform Verification of Parameterized Systems
    - Deadlock freedom
    - Transitive non-interference

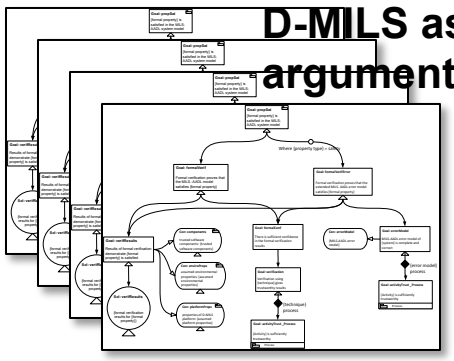
Compositional Verification Tool Chain



WP4 Compositional Assurance

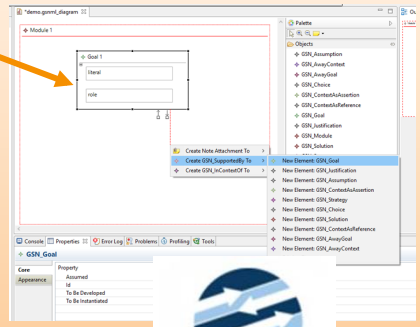
- Prior to D-MILS
  - Manual construction of assurance case
  - Manual confirmation of completeness and correspondence to system design
- Accomplishments of D-MILS
  - Assurance argument pattern catalogue
  - Automation of assurance argument creation with pattern instantiation from models by Model-Based Assurance Case (M-BAC) tool
  - Integration of assurance argument with design and verification by Weaving model with explicit mapping to system models

Compositional Assurance Tool Chain



D-MILS assurance argument patterns

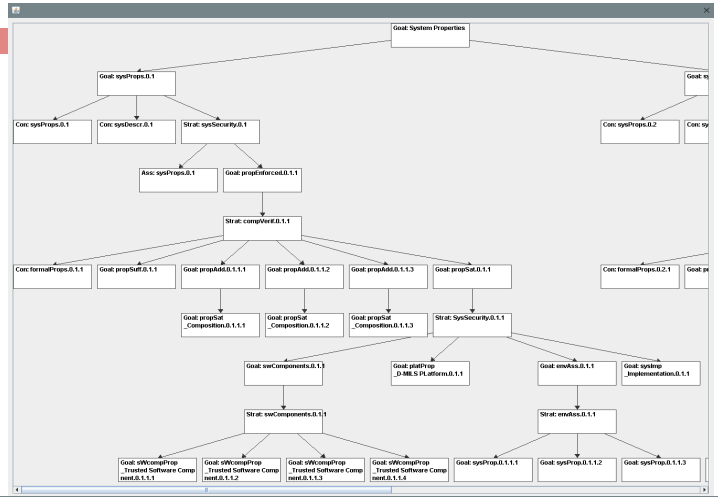
M-BAC pattern editor

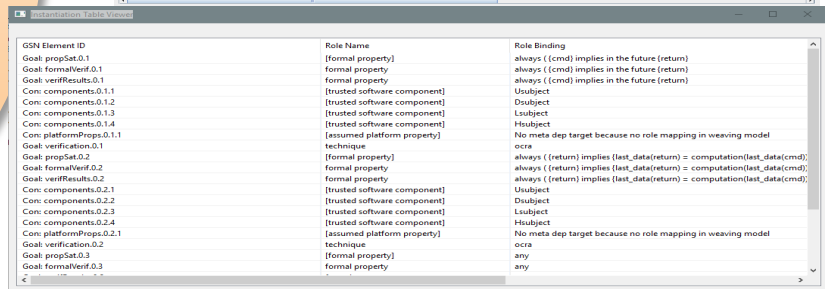


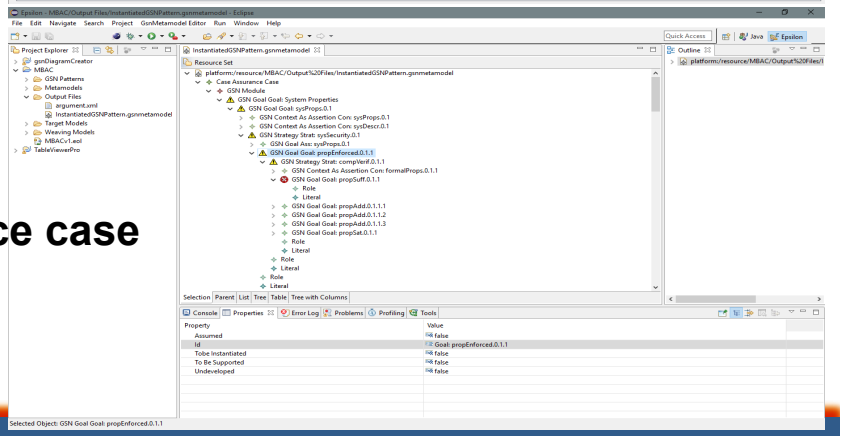
M-BAC

Weaving model

D-MILS system models







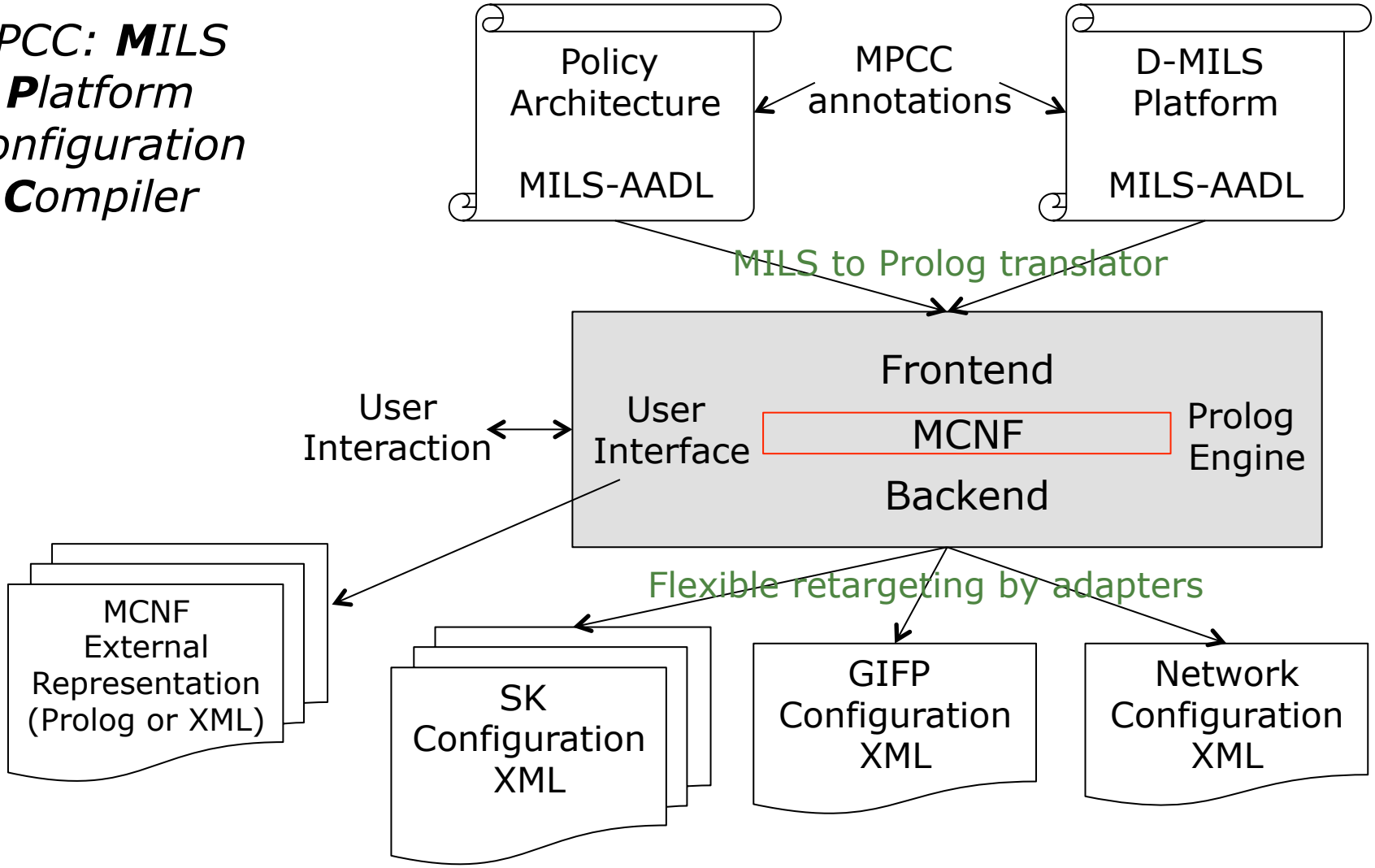
D-MILS assurance case

WP5 MILS Platform Configuration Compiler

- Prior to D-MILS
  - Single system, no distributed or global view
  - No interoperability of diverse products
  - No cohesive implementation of "MILS Platform"
  - No traceability of deployment to design spec
- Accomplishments of D-MILS
  - Global configuration of distributed MILS system
  - Flows from high-level design specification
  - Flexible targeting based on adapters
  - Configuration representation standard emerging
  - Configuration correctness validation

Configuration Compiler

*MPCC: **M**ILS **P**latform **C**onfiguration **C**ompiler*



WP6 D-MILS Platform - SK extensions

- Prior to D-MILS
  - Independent separation kernel implementation and standalone configuration
  - "D-MILS Platform" as an integration of SK, MILS Network System, and MILS Console was only a concept (MNS and MCS did not previously exist)
- Accomplishments of D-MILS
  - Distributed MILS Platform from integration of LynxSecure and MNS/TTEthernet + MCS
  - Security extensions to TTEthernet
  - Deterministic system: nodes ***and*** communication
  - Operational support for distributed architectures
  - Platform configuration tools now connected through D-MILS tool chain all the way back to the MILS-AADL system model

WP7 Industrial Demonstrations

- Detailed evaluations performed
- Frequentis Voice Service (FVS)
  - MILS AADL specs for FVS
  - Prototype MILS Console system
- fortiss Smart Micro Grid (SMG)
  - MILS AADL specs for SMG
  - Starlight example – MILS-AADL to configuration to running demo
- Provide excellent feedback for future development of D-MILS tools/platform

Summary and Lessons Learned

- Our goals were ambitious and we are very proud of our accomplishments.
- We've learned much about what needs to be done to improve usability and applicability of tools and platform
  - Improvements for platform performance
  - Integration, modularity, and UI of tools
  - Methodology and supporting features
  - Extension to dynamic systems
  - Upgrade TRL of all components