Open Discussion #2

LISBON
2025

Goals Processing in Agentic Frameworks

Coordinators

Dr. Steve Chan, VTIRL, VT, USA

Prof. Dr. Petre Dini, IARIA, USA/EU

Statement: We were all using an Agentic framework, with one small caveat: not based on a natural language narrative for agent communications, but based on formal rules and well-defined protocols, instead.

Stories: I remember when I was presenting the project requirements and goals, while some people were already starting to write the code. I recall that with partial accuracy only, we achieved 99.999 service availability across the USA.

Open Discussion

LISBON
2025

Basis: requirements, players & duties (hardware and software), message flowcharts, hardware selection, software framework, APIs, Interfaces, user interfaces, etc. [then: verification, validation, assurance)

MODEL

Abstract classes, objects, aggregation, inheritance, object contracts, agents (smart objects), MAS, agentic framework

Keywords of change: automation, self-reasoning, and self-healing

'Old' OO (or so) models

> English narrative
> Structured requirements
> Nouns, verbs -> objects, actions, **goals**
> OO-framework (IBM: Java, Eclipse}
Framework objects (cca 10%)
Specific objects, operational request
Middleware for object communication (traders, brokers, hierarchy, etc.)
BUS architecture (event subscription)
Object storage (specific databases: ObjectStore, etc.)

Formalisms
Verification
Validation
Maintenance
Modeling
Simulation
Monitoring
Management
Reflective architectures
vs
Digital Twins

'New' Agentic frameworks

> English narrative
> Structured requirements
> LLCs identify main requirements
> LLMs identify specific constraints and **goals**
> Agentic-framework
Framework agents (cca 90%)
Specific agents
Middleware as an agent [Orchestrator]
Communicating Agents (hallucinations, bias)
Agents library

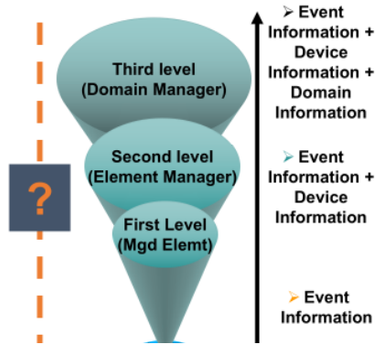
Industrial Challenges in Working with Events

Prof. Dr. Petre DINI
Senior Technical Leader, NMTG Office of the CTO
Cisco Systems, Inc.
pdini@cisco.com

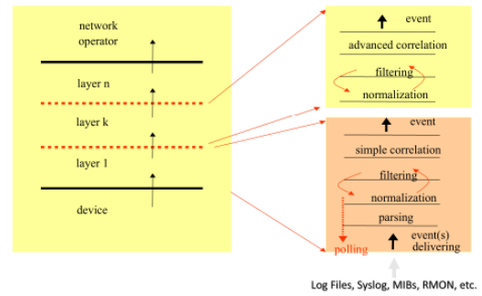
DEBS 2004
Edinburg

Bottom-up vs. Top-down

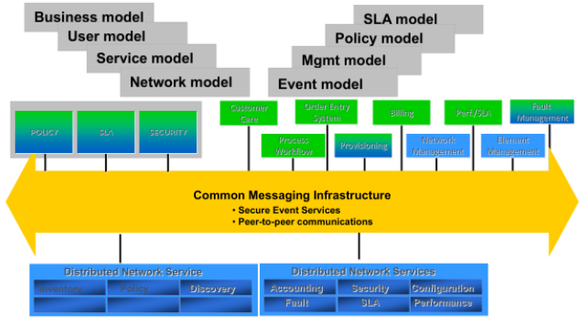
- Domain Manager enriches with domain information
- EMS enriches with multi-device information
- Notification Engine collects OS notifications



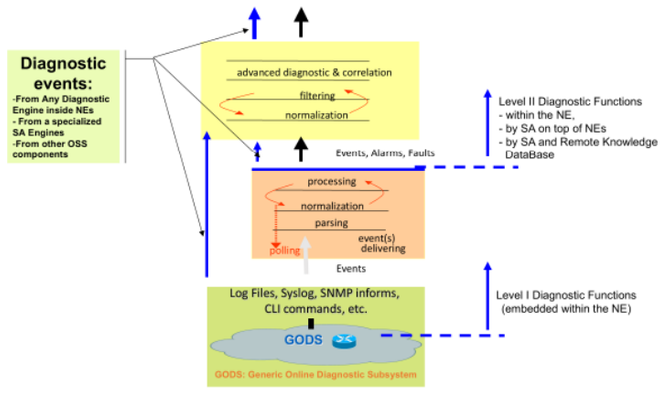
A Layered Processing View



Communication Bus

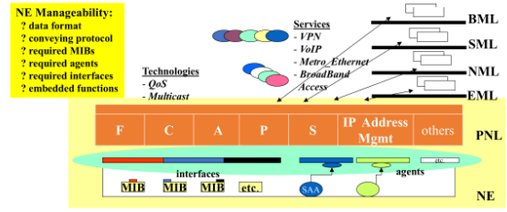


Multi-level diagnostic

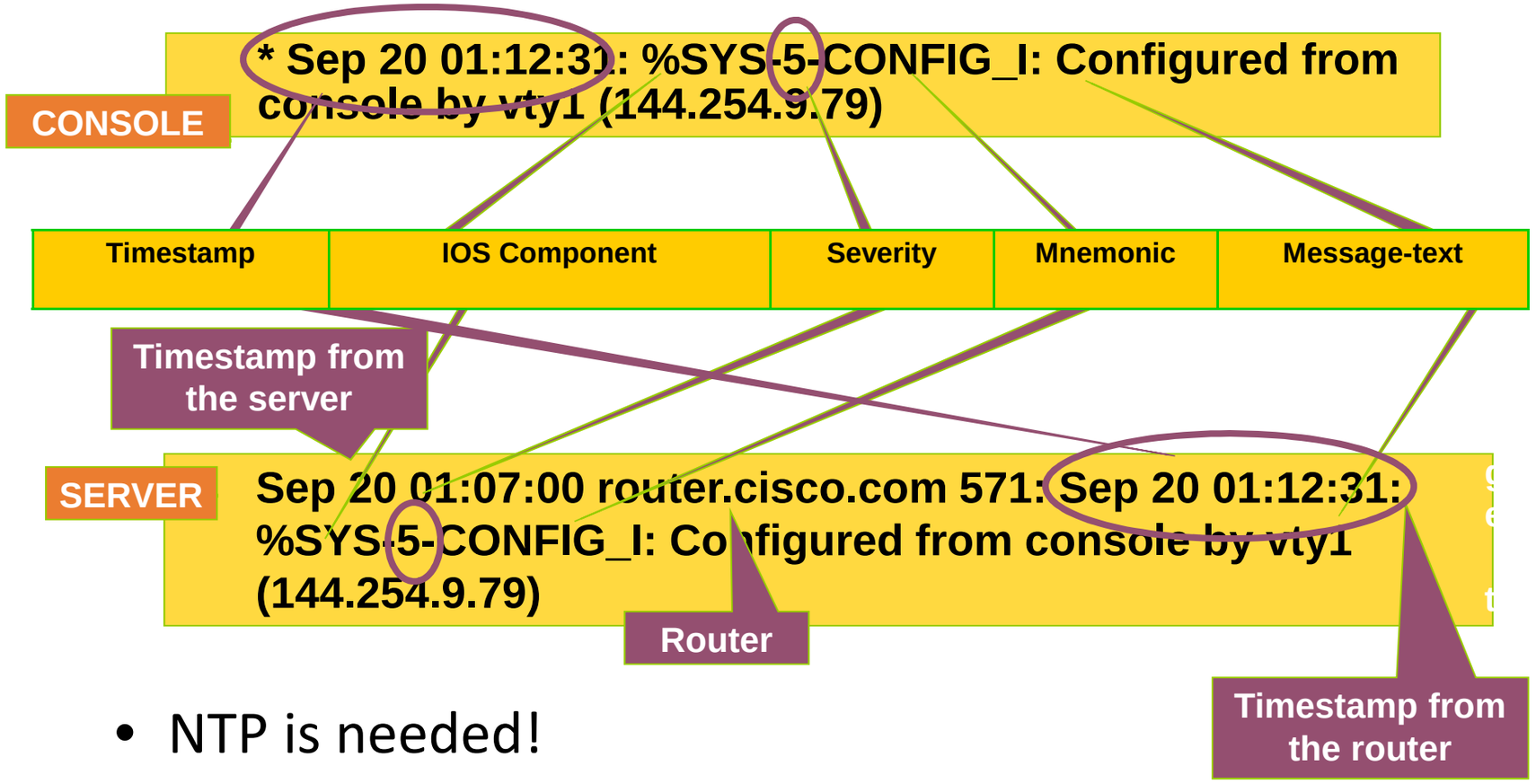


Syntax Issues

- Various formats
- Myriad of conversions needed
- Lack of syntax control



Syslog Message “Body” Format in the IOS



- NTP is needed!
- **Header:level can be different than Body:severity**

Open Discussion

LISBON
2025

(Example) Assume

a. ***We have a paragraphs of ten sentences describing a potential system (irrelevant, but as an example: selling football tickets, coming with requirements for distribution, security, authenticity of the tickets, etc.)***
b. ***We opt for using an agentic framework***
c. **Q1: What are the sequence of steps?**
d. **Q2: Are there dedicated agents we should choose from (having definite roles, limitations, etc.)?**
e. **Q3: Can we build personalized agents and insert them into the whole framework?**
f. **Q4: How are the constraints of the original system (requirements) translated into the goals of the agentic framework and how are the goals assigned to the agents, namely, are they split (par., seq.) in sub-goals, or joint, or mediated if they are conflicting?**

Note: goals conflicts can be static (easy verifiable) or dynamic (changing, status, volatile, …)

Q1 — Sequence of steps (agentic design pipeline)

- **Requirements → Constraints**
  Normalize the paragraph into atomic constraints (e.g., “only 6 tickets/user,” “cryptographic authenticity,” “latency < 200 ms,” “fair queueing,” “GDPR compliance”).
- **Constraints → Goals** (WHAT)
  Translate each constraint into one or more goals with acceptance criteria (e.g., *VerifyTicketAuth* with “<5 ms verify; FIPS-approved crypto”).
- **Goals → Capabilities** (HOW)
  For each goal, list capabilities needed: verify signature, allocate inventory, anti-bot scoring, payment, ID verification, audit logging, anomaly detection.
- **Organization model**
  Choose agent types and interaction styles: hierarchical (coordinator), market/contract-net, or peer mesh. Define authority, priorities, SLAs.
- **Allocation**
  Map goals ↔ agents via: required capability match, trust level, performance budget, and data locality. (Greedy first fit → refine with constraints solver).
- **Coordination protocols**
  Pick protocols per interaction: request–response, publish/subscribe, contract-net (bids), two-phase commit, saga (compensations).
- **Conflict handling**
  Predefine policies for scarce inventory, double-spend, identity disputes, fairness vs. revenue, security vs. latency. Attach tie-breakers.
- **Assurance hooks**
  Add runtime monitors (temporal rules), guard rails (Simplex/shields), provenance logs, and canary scenarios.
- **Simulation & dry-runs**
  Load/chaos tests with adversaries (scalpers/bots), failure injection, latency budgets.
- **Deployment with continuous governance**
  SLAs, rate limits, ABAC/RBAC, rotation of keys/models, drift control, post-mortems.

Q2 — Dedicated agents (typical roles & limits)

- **Orchestrator/Goal Manager**: decomposes goals, assigns tasks; limits: no direct data custody.
- **Inventory Agent**: seat allocation, holds, releases; limits: cannot bypass fairness policy.
- **AuthN/Z Agent**: KYC/ID checks, RBAC/ABAC decisions; limits: no pricing authority.
- **Crypto/Attestation Agent**: signing, verification, key rotation, HSM access; limits: read-only to PII.
- **Payment & Risk Agent**: PSP integration, fraud scoring, SCA, chargeback handling; limits: cannot allocate seats.
- **Anti-Bot/Trust Agent**: device fingerprinting, rate-limit advice, CAPTCHA orchestration; limits: advisory → Orchestrator enforces.
- **Queueing/Fairness Agent**: virtual lobby, lottery/queue discipline, per-user caps; limits: cannot edit ticket metadata.
- **Compliance & Privacy Agent**: data minimization, consent, retention, audit trails; veto power on unlawful flows.
- **Observability Agent**: SLO monitors, tracing, anomaly alerts; limits: no business decisions.
- **Settlement & Ledger Agent**: immutable log (append-only), refunds, compensations; limits: no user policy changes.

Q3 — Personalized agents

Yes. Define a capability contract (inputs/outputs, pre/post-conditions, latency & trust class), implement your agent, and register it with the **Orchestrator**. It can then be selected during allocation if it satisfies:

```
capabilities ⊇ goal.reqs && SLA_met && policy_compliant && trust_level_ok
```

Translating Requirements → Goals → Agents (mini example)

1. Requirements (excerpt)

R1 Authentic tickets only; cryptographic validation.
R2 Max 6 tickets/user; prevent bots.
R3 Fair access at drop time; no cart hoarding.
R4 End-to-end latency < 200 ms.
R5 GDPR compliance; immutable audit.

2. Goals

G1 *VerifyTicketAuth* (verify ≤5 ms, FIPS algos).
G2 *EnforceUserCaps* (≤6/user, per-event).
G3 *EnsureFairAccess* (virtual lobby + lottery/queue).
G4 *MeetLatencyBudget* (<200 ms, back-pressure).
G5 *Provenance&Audit* (append-only, replayable).
G6 *AntiBotMitigation* (risk score; action ladder).
G7 *PrivacyCompliance* (min data, DSR support).

3. Allocation (sample):

G1 → Crypto/Attestation Agent
G2 → Queueing/Fairness Agent + AuthZ Agent
G3 → Queueing/Fairness Agent (lottery/queue policy)
G4 → Orchestrator + Observability (shed/back-pressure)
G5 → Settlement & Ledger Agent
G6 → Anti-Bot/Trust Agent (+ Orchestrator enforcer)
G7 → Compliance & Privacy Agent

4. Conflict patterns & policies

Fairness vs. Revenue (R2 vs dynamic pricing): declare **lexicographic priority**: safety/security → compliance → fairness → revenue.
Latency vs. Security (R4 vs strong checks): apply **progressive trust**: light check on hot path; deep check async or on anomalies.
User Cap vs. Group Orders: introduce **goal refinement**: *EnforceUserCaps* → per-identity + per-payment-instrument + per-device.
Anti-Bot false positives vs. Fairness: dual-channel appeal (human-in-the-loop) with bounded SLA.

Open Discussion

LISBON
2025

History revisited (i)

- Requirements
- Requirements tracability
- Pre-post conditions
- Control policies (Definition/Access Points)
- Agent contract agreements
- SLA/SLO agreement
- Formal specification of interactions (V&V)
- (Formal Robust Protocols)
- Unique standard framework (s) (Eclipse, as an example)
- Patterns, Artefacts, Software reuse
- Customized (embedded) agents
- Formal agent communication (trusted exchanges)

History revisited (II)

SLA/SLO specifications
UML (semi-formal) specification (another tens, or hundreds)
SDL, LOTOS – protocol formal specifications
Patterns Catalogues
Policy Formal Definitions/Frameworks (type, actions, guarantees)
Activities: actions, plans (par. & seq., actions, temporal aspects, conflicts, mitigation, etc.)
Versioning control (configuration mgmt)
Support for Legacy systems
! 99,999 service availability

Q; Status quo

Design time consuming
Limited knowledge (of some, all)
Human-depending productivity
High skilled experts (cost)
Long learning curve
Poor code documentation / manuals

Formal Methodologies

Rebecca Wirfs-Brock - Responsibility-Driven Design (RDD) (OOPSA 1989+)
Bertrand Meyer - Design by Contract (DbC) - Eiffel programming language (~ 1986 +)
Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides,
with a foreword by Grady Booch: - Design Patterns (reusable elements) (OOPSLA, 1994)

Agents

AT&T - Monitoring and Management system had ~ 600 specialized agents (~2000)
Cisco Systems - inside each router (~50 agents, fault, performance, etc.)
ODP (1990 - Trader - formal definition), CORBA (1990 Broker - SDL specifications), TINA, etc. (Manager) --- > Agentic (Orchestrator).

BACK TO AGENTIC FRAMEWORK (again LLMs/LCMs and some standard agents)

Q: Status quo

Design time consuming
Limited knowledge
Human-depending productivity
High skilled experts (cost)
Long learning curve
Poor (code) documentation / product manuals

A+

Quick design (more than Agile approach)
Prompt information at large scale [caveat-pre-knowledge is needed]
Automation-based productivity (less human workforce0
Min high skilled exerts (prompt experts and tools knowledgeable)
Long learning curve (almost instant; see prerequisites)
Instant generation of documentation / manuals

A- (to be improved)

Deskilling
Highly depending on a few individuals
Lack of or not at a required level of Explainability, Ethics(Opaqueness)
Uncontrolled bias (European Act, USA ….)
Great ROI (for some)
Unreliable information (hallucinations, unintended (or not) consequences,
Biased, unreliable and not trustable communications between agent

A-/+ (to be improved)

Decision of NLP is not accurate (see Syslog payload field)
Difficult cu catch errors/mistakes
Bias in data sets (V&V)
AI literacy, Data literacy

Open Discussion

We are here, Agentic frameworks are here, too!

QUO VADIS?
Rolling up the sleeves!

STAGE IS YOURS