Rune Technologies – Technical Diligence Agenda

Attendees (SF): Jim Wiese, Mitch Christensen, Azeem Feroz, Andrew Randall, Keith Wilson, Daniel Lim, Gary Wang, Cole Cary, Graham Pinto, Aaron Huang

---

A. End-User Applications and Platform (TyrOS)

- Full module inventory: Which of the following are GA, beta, or roadmap
  - Predictive logistics (ammo, fuel, parts forecasting)
  - Distribution scheduling (air, ground, maritime transport coordination)
  - Predictive maintenance (fault forecasting, parts mapping, readiness scoring)
  - Medical logistics (casualty forecasting, CASEVAC/MEDEVAC routing, treatment capacity)
  - Logistics UxV orchestration (autonomous resupply tasking across domains)
  - Host-nation contracting (local vendor integration, additive manufacturing)
  - Vehicle sensor integration (telemetry ingestion, fuel/ammo/fault data)
  - Others?
- Provide overview of the components of the TyrOS platform - for each, what is the underlying technical component, what is proprietary 1st party developed vs open source / COTS? Component specific questions below:
  - Meta - how is the ontology generated (eg FDE populated?)
  - Norn -
  - Router
  - Tracker
  - Forecaster
  - Optimizer

    - Raido
    - Planner
- Data platform / backend
    - Is TyrOS's data structured – entities with stable identifiers we can reference from agentic workflows or opaque text?
    - Overview of data model for Meta and Tracker platform components

B. Agent and AI model layer

- Walk through the technical components of Saga
- LLM model architecture:
    - Which underlying LLM(s) power Saga's reasoning?
    - What prevents Saga from generating a Course of Action that's dangerous, wrong, or hallucinated (guardrails)?
    - How do new/updated models or Saga logic reach edge nodes running on 5 kbps mesh, where weights are gigabytes but the pipe carries operational traffic?
    - Hallucination / error rate
    - Attribution mapping mechanisms
    - Compute overhead for full traceability of model outputs to data, context, model parameters (if available)
- Predictive AI model architecture and provenance:
    - Forecaster: Classical ML models - ARIMA/SARIMA, Long short term memory (LSTM), XGBoost
    - Optimizer: Linear programming, mixed-integrtion programming, other algorithms, use of RL, simulated annealing, etc
- Where inference runs: on-device at edge vs. cloud-dependent
    - Model distribution and update path – full weights, delta updates, OTA via mesh, sneakernet? Cadence and operator control?
    - If inference runs on-device at the edge, what adaptation capabilities exist locally – LoRA / PEFT fine-tuning on locally-cached data, in-context learning from recent operator interactions, or none?
- Training data:

    - Request data lineage and volume for each source of training data
    - How much of the data for Rune AI models is customer-specific vs public or synthetic?
- Agent architecture - grounding, evaluation/observability, data ingest, RAG
- AI / agent evaluation
    - How does Rune measure whether Saga's recommendations are correct, sound, physically feasible, and mission-appropriate?

C. Infrastructure and Edge Architecture

- Offline-first data model: what lives at the edge vs. cloud broken down by module
- Conflict resolution on reconnect (CRDT, last-write-wins, custom merge)
    - Is the conflict resolution strategy a global setting or specified per module, entity, etc.?
    - CRDT layer - custom or 3P? CRDT scalability
    - 5 kbps + mesh + auto-conflict-resolution – Has this been proven at scale (~15K+ personnel, hundreds-to-thousands of nodes) with some isolated for days, then reconnecting cleanly – in real operations?
- Minimum hardware footprint per node (Toughbook vs. server-class)
    - Are there minimum specs?
- Mesh networking: proprietary or existing protocol (MANET, TSN), state synchronization
    - Is multi-hop relay supported?
    - Does this use a proprietary protocol or layer over existing transports?
    - What is the replication model (i.e. does every node hold every record? Hierarchical?)?
    - Under resource contention, what gets dropped first?
    - Reconnection SLAs: time to eventual consistency after reconnecting?
- Infra hosting architecture for each customer deployment of Rune

D. Integration Surface

API surface and integration patterns

- Client library or raw protocol/API (both?)?
- Supported transports (REST polling, gRPC streaming, custom)?
- Where does the integration run (TyrOS, gateway, cloud)?

TyrOS ADK & Extensibility

- TyrOS autonomy development kit: functionality, architecture
- Are the 'Autonomy Development Kit' (ADK) and the data-ingest SDK one artifact or two?
- Extensibility model (plugin architecture, custom module development)

External System Integrations

- Lattice/ADVANA/Maven integration depth: bidirectional or read-only
- ERP interface (GCSS-Army, GCSS-MC)
- Published schema/entity model for sustainment objects

E. Security and Accreditation

- Confirmation of IL5 and IL6 ATOs (authorizing agency / provenance)
- Encryption and key management in disconnected environments
    - Is encryption at rest on the edge node supported?
    - How does an off-line node receive a new key on key rotation?
- RBAC, ABAC

F. Deployment Maturity

- Which deployments are contracted production vs. experimentation?
- Time from contract to operational capability
- Deployment model - FDEs or can customers operate themselves?
- Scalability ceilings:
    - # of vehicles? # of sensors? What are the scale bottlenecks?

G: Engineering processes

- SDLC & Development Model
    - SDLC process today and alignment with FDE model
    - How are customer requirements and custom builds for customers fed into the core product?
- Quality & Tech Debt
    - QA and testing; code coverage metrics
    - Tech debt / engineering modernization efforts
- Security & Compliance
    - Vulnerability management; open vulns today
    - SBOM / artifact signing
    - Open source code usage and licensing
- IP & Third-Party Dependencies
    - Reliant on IP from 3Ps (what, how critical, what's the fallback?)
    - Licensing exposure (copyleft contamination, commercial restrictions)

Priority Artifacts to show in DD

1. Live demo showing disconnected operation and reconnection
2. Module-by-module capability maturity matrix (GA vs. beta vs. roadmap)
3. Architecture diagram (edge/cloud topology, data flow)
4. AI model (predictive AI, generative AI) architecture, and AI agent architecture