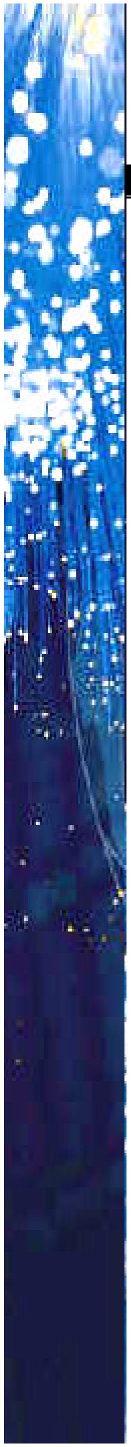
Integrating Soar into the OneSAF Models Framework

Dr. Doug Reece

SAIC®
Science Applications
International Corporation
An Employee-Owned Company

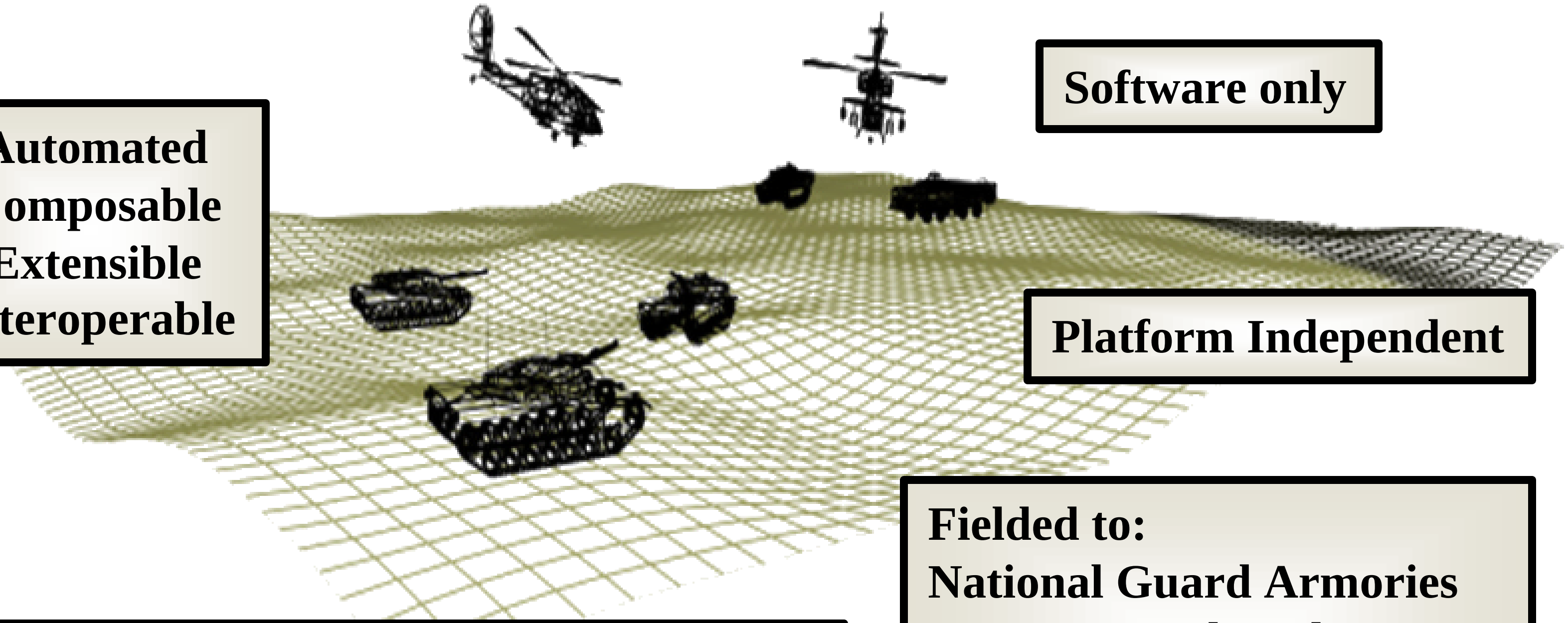


Outline

- OneSAF Modeling Infrastructure (MI) overview
  - Emphasis on entity architecture
- Ideas for integrating Soar
  - Not *interfacing*

OneSAF Summary

A composable, next generation Computer-Generated Forces (CGF) that can represent a full range of operations, systems, and control process (TTP) from entity up to brigade level, with variable level of fidelity that supports multiple Army Modeling and Simulation (M&S) domains (ACR, RDA, TEMO) applications.



Automated
Composable
Extensible
Interoperable

Software only

Platform Independent

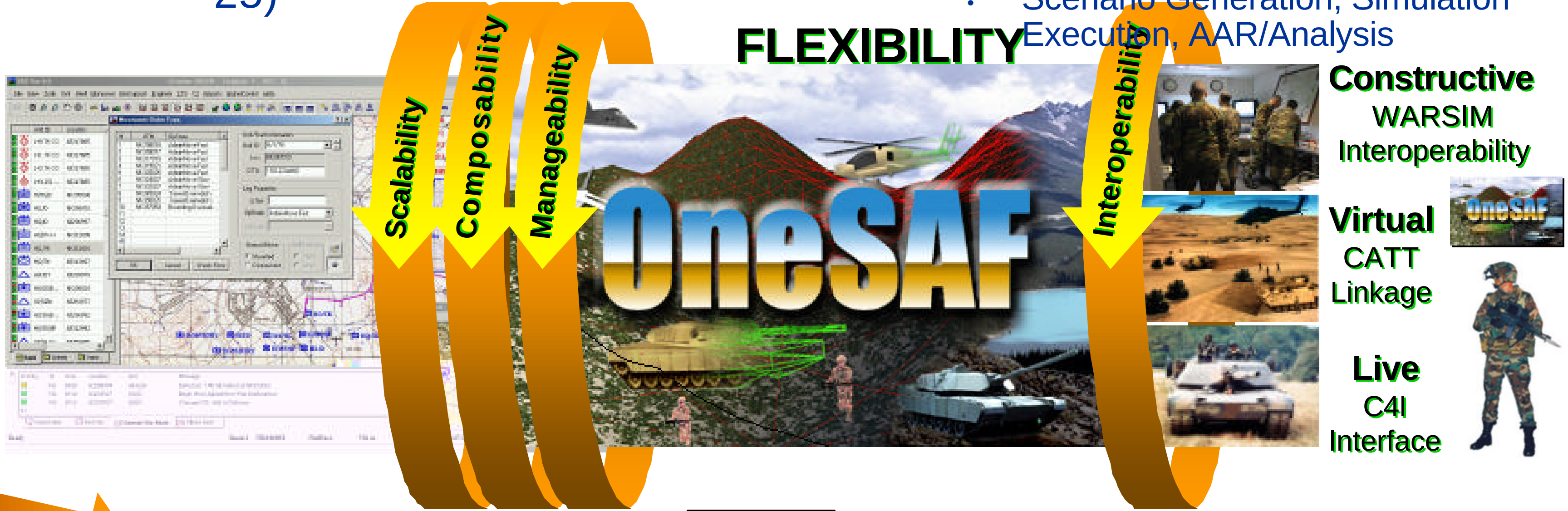
Replaces legacy entity based Simulations: BBS - ModSAF - JANUS - CCTT SAF – AVCATT SAF

Fielded to:
National Guard Armories
RDECs / Battle Labs
Reserve Training Centers
All Active Duty Brigades and Battalions

OneSAF Requirements

- Interoperability (HLA/DIS, WARSIM, CATT, and two-way, real-world C4I)
- Entity-based model that can display collective unit icons
- Composable equipment and units (up to battalion size)
- Complex automated behaviors
- Multiple sides (a minimum of 25)

- User-friendly features (scalability, composability, and manageability)
- Dynamic data editors
- 2-D and 3-D visualization
- Data collection, analysis, and AAR graphics and tables
- Compatible with various hardware (WARSIM and PCs)
- Full Lifecycle
  - Scenario Generation, Simulation Execution, AAR/Analysis



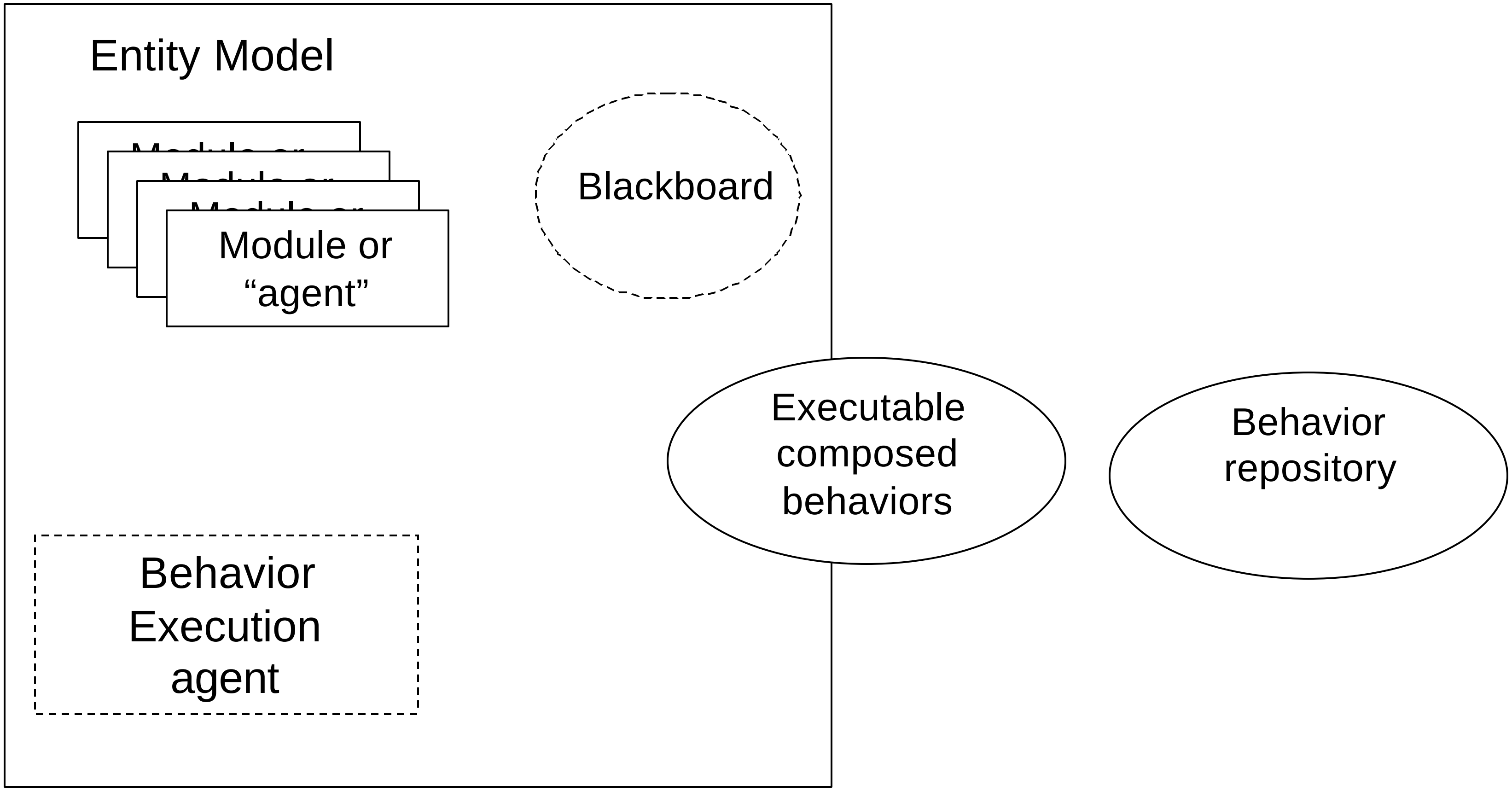
Manageability

Data Editors: unit, organization, behavior, terrain, environment, battlefield graphics, user preferences, fire support, engineering, damage and equipment repair, logistics

MI Goals

- Just the infrastructure
  - Behavior “content” comes later this year
- Represent normative military behavior
- Represent reactions to events
- SME-composition of behaviors (!)
- Software Engineer definition of
  - “primitive” behaviors
  - Predicate functions
- So once the primitives are done, the SMEs will program all the behavior…

Entity Overview



Entity Model

Module or “agent”

Blackboard

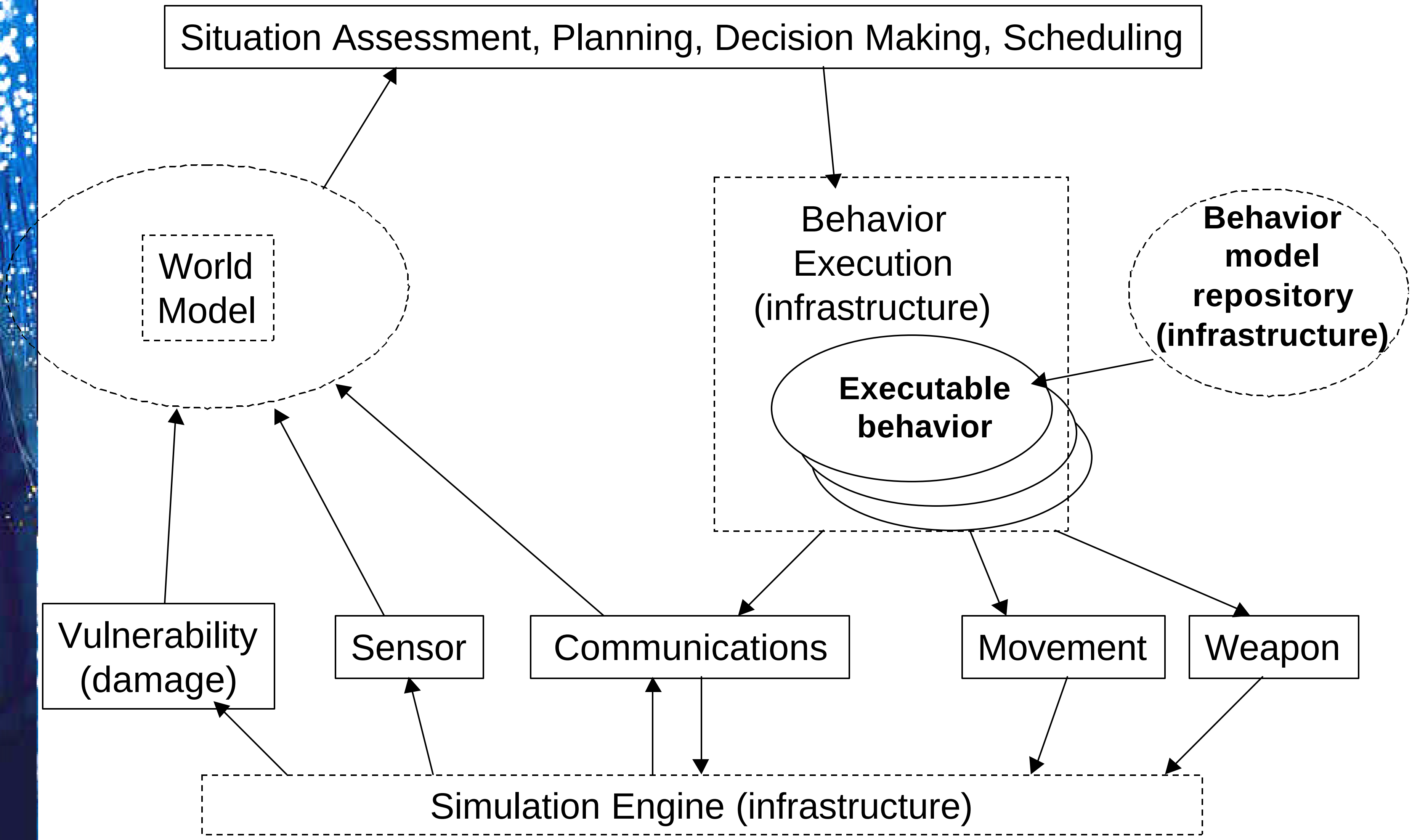
Behavior Execution agent

Executable composed behaviors

Behavior repository

Entity description language defines entity modules

Entity Model



Situation Assessment, Planning, Decision Making, Scheduling
World Model
Behavior Execution (infrastructure)
Behavior model repository (infrastructure)
Executable behavior
Vulnerability (damage)
Sensor
Communications
Movement
Weapon
Simulation Engine (infrastructure)

Blackboard

- All entity modules (agents) communicate via BB
- Modules must register to receive data
  - Register for a class of data
  - Detonations, communications, clock timers…
- Execution is event driven
  - On new event (data), BB executes all registrants
  - New events (data) produced by modules causes second wave of execution
  - Runs to quiescence
- Events do not persist on BB

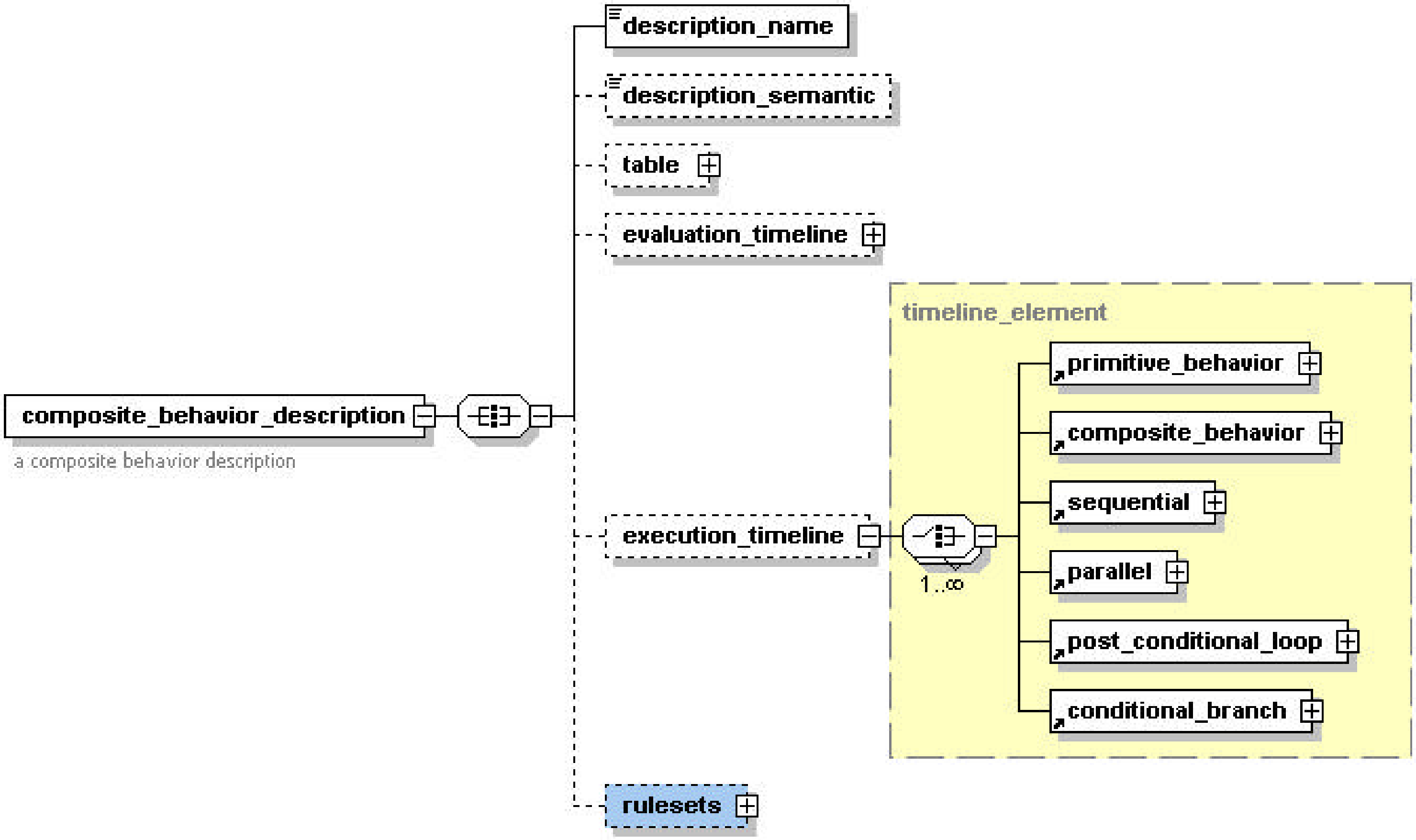
World Model

- Stores perceptions, inferred facts
- Facts are instances of fact-class objects
  - Pre-defined set of objects and attributes
  - Rules in behavior models can assert or retract facts
- Pre-defined set of predicates and functions
  - Rules combine predicates with boolean ops
  - Matching may be limited (or not)
    - E.g. $\forall(x)$ (IF Entity(x) $\wedge$ Tank(x) $\wedge$ (Range(x) $<$ 2000)) may be implemented by walking a known entity list

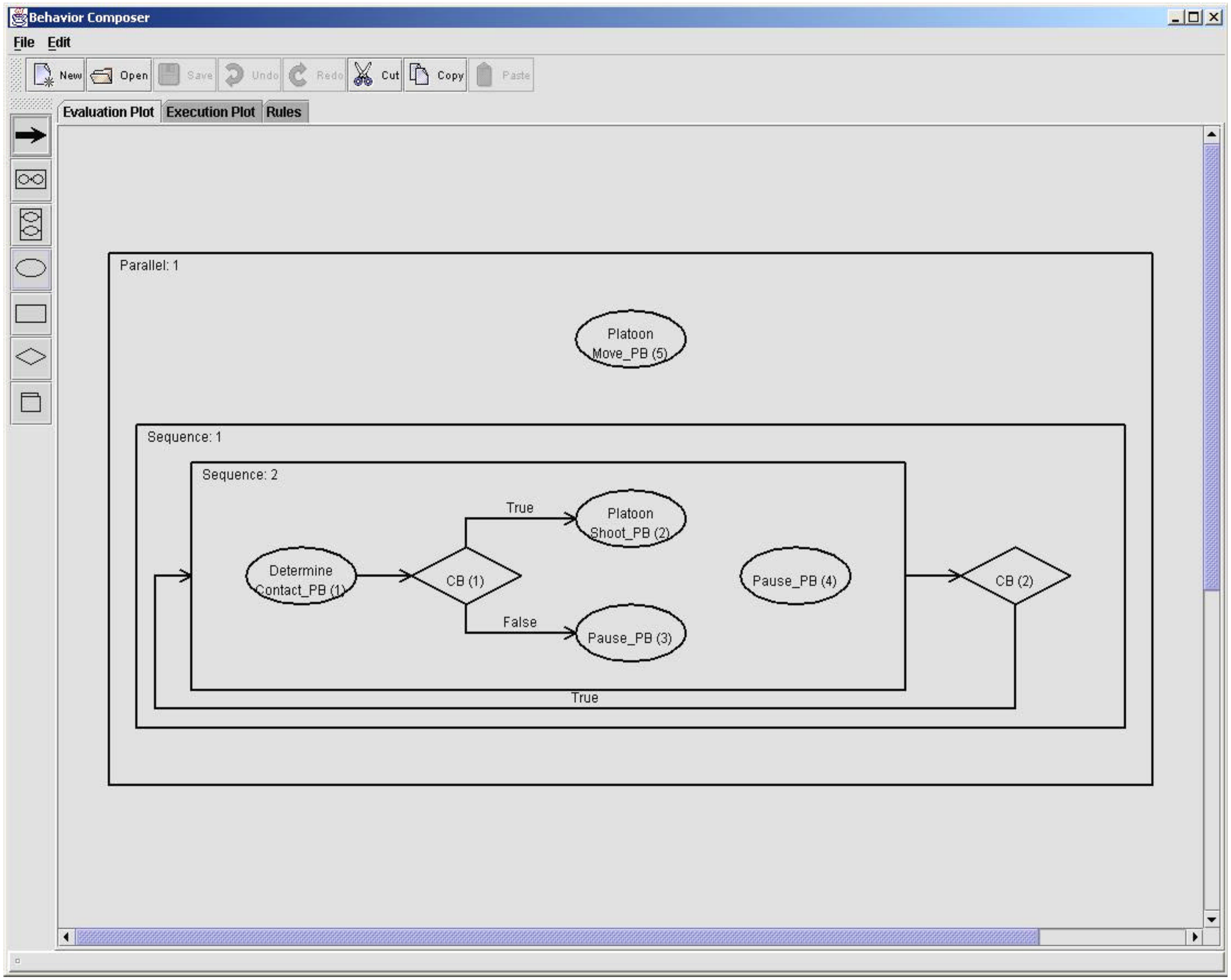
Behavior Models

- Flow chart-like action sequence descriptions
  - Includes sequential, parallel, looping, and branching
  - Use predicates to branch
- Actions can use parameters
  - Parameters can be computed by preceeding actions
- Actions can be primitive or composite
- Include IF-THEN rules
  - Assert facts or
  - Start reactive behaviors
    - Interaction between reactive and background behaviors still undefined

Behavior Model Language Structure



Behavior Modeling Editor



Primitive Behaviors

- Simple actions using physical agents in entity
  - Put command data on blackboard, which triggers physical agents, OR
  - Calls physical agent functions directly
- Process data
  - make inferences
  - Add facts to world model

Soar

- Rule-based engine +
- Facts
- Rules
- Goal stack

Facts

- Arbitrary object-attribute-value triples
- Working memory
- Special facts for input and output

Rules

- General matcher
  - Modified RETE for efficiency
- Long-term memory
- Knowledge of preferences for taking action in certain situations

Soar Agent

- Interface code puts input data into facts
- New data may trigger rules
  - Change decision anywhere in goal stack
    - Make new subgoals from that point
  - Make inferences outside of goal stack
  - Put new data in output facts
- Runs to quiescence

Approach for Soar in OneSAF

- Replace all behavior agents with Soar
- **Use OneSAF physical agents**
  - Primitive actions only
- Replace blackboard (Soar working memory)
- Replace world model (Soar working memory)
- **Use OneSAF composed behaviors**
  - But don’t execute in OneSAF infrastructure

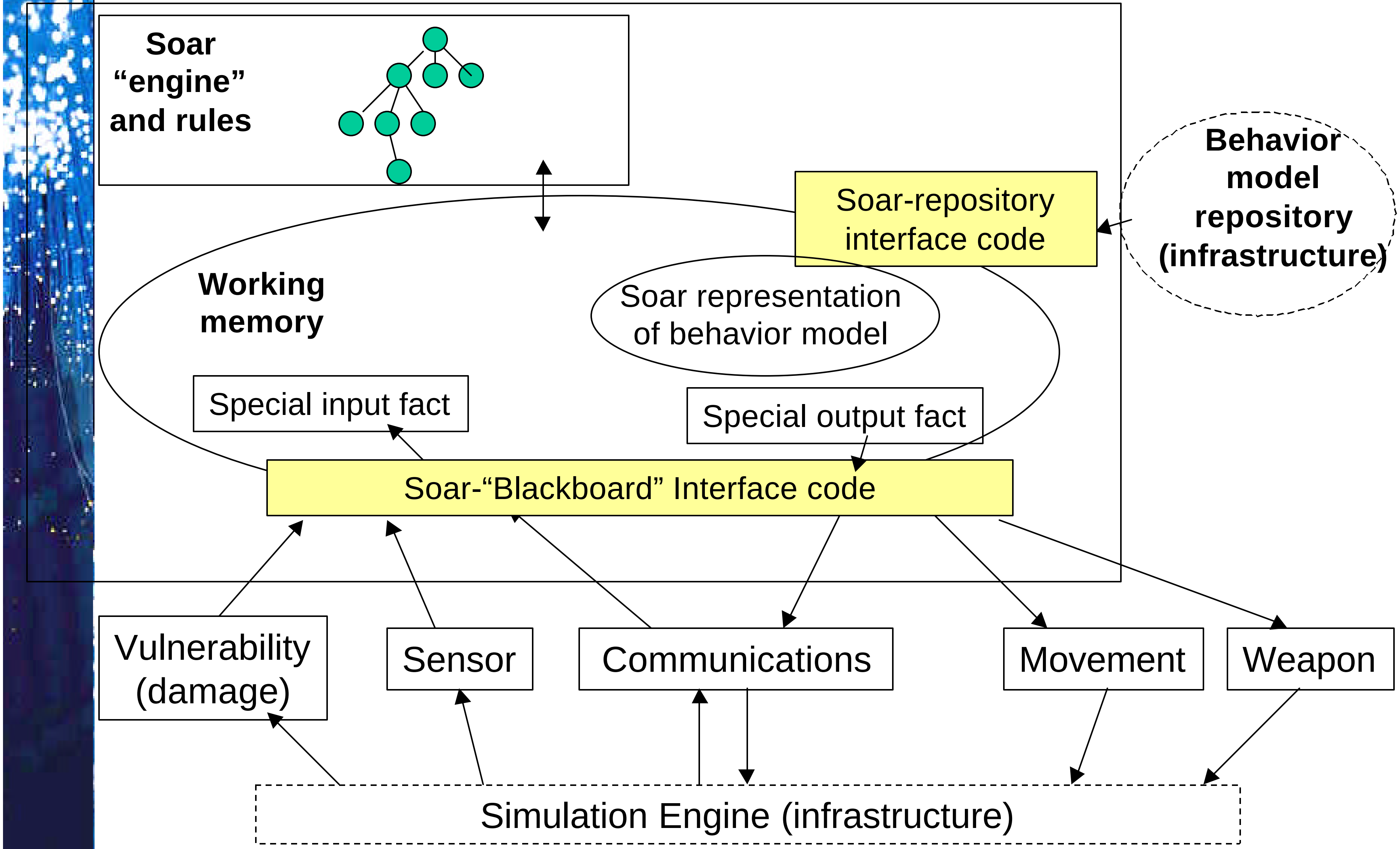
Replace Blackboard…

- Create interface between physical agents and Soar using OneSAF BB protocol
  - Allows same physical agents to be used
  - Interface puts input in working memory, sets triggers from working memory

Use Composed Behaviors…

- Soar reads composed behaviors from OneSAF run-time repository
- Soar builds its own representation of order/behaviors
- **Soar must implement the same named primitives used in behaviors**
  - Named facts
  - Predicate functions
  - Primitive behaviors

Soar Entity Model Example


Soar "engine" and rules
Behavior model repository (infrastructure)
Soar-repository interface code
Working memory
Soar representation of behavior model
Special input fact
Special output fact
Soar-"Blackboard" Interface code
Vulnerability (damage)
Sensor
Communications
Movement
Weapon
Simulation Engine (infrastructure)

Conclusion

- Interesting thought experiment
  - …for now…
- Ideas? Simple examples?
  - Interfaces
  - Soar representation of flow-chart behavior