The Behavior Analysis Engine
Context

EMIR
Engineering Models
Information Repository

AE
Analysis Engine

AUI
Analysis User Interface
Context

AE and Reasoning Engines (e.g., Aspen)

Operational Data from:
- Test Beds,
- Simulated Scenarios,
- Live SGRDP systems

Configuration Data

Analysis User Interface

- DDR Placement
- Mitigation Effectiveness
- Attack Visualization

Fault/Attack Trees

Information Exchange

Goals

Architecture Artifacts

Engineering Models & Information Repository

Functions & Flows

Hardware & Software Inventories

Communications

HWR/SWR deployment
Comparing what we have and what we planned with related work

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Features/Criteria</th>
<th>Continuous variables</th>
<th>Time</th>
<th>Content/Change</th>
<th>Specific problems</th>
<th>Non-linear problems</th>
<th>Object-oriented</th>
<th>High/Fidelity</th>
<th>Usability</th>
<th>Execution</th>
<th>Verification &amp; Validation</th>
<th>Adaptability/learning</th>
<th>Simulators</th>
<th>Tools &amp; Industry analysis</th>
<th>Testing &amp; Troubleshooting</th>
<th>Research</th>
<th>Problem Solvers</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Modeling</td>
<td>MagikDraw + plugins</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Current All-MagikDraw &amp; AADL + ADL</td>
<td>yes</td>
<td>user must code</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Current All-MagikDraw &amp; ADL + AADL + ADL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>System modeling &amp; analysis tools</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Phoenix Technologies</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ptolemy II + PTIDES</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AADL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulators</td>
<td>CS and CT support</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>V&amp;V distributed</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gates/Industry analysis tools</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COTS security tools</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Research</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Problem Solvers</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Learning tools</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>statistical sampling</td>
<td>yes</td>
<td>activity &amp; diagnosis</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Execution**: yes
- **Verification & Validation**: yes
- **Adaptability/learning**: yes
- **Simulators**: yes
- **Tools & Industry analysis**: yes
- **Testing & Troubleshooting**: yes
- **Research**: yes
- **Problem Solvers**: yes
- **Learning tools**: yes
How is the AE approach different?

1. More expressive modeling language
   - simulators (e.g. Simulink) don’t solve problems
   - problem solver models often lack fidelity from lack of the language features
     • object-oriented class structures
     • continuous variables, time, state change
     • a variety of operators/functions (some just support logic)
     • quantification ("all computers connected to the public wireless network")
   - The current AE supports these (and uncertainty to a limited degree)
   - Language enables dynamic creation of constraints
     (others that do this: MDS, some automated planners, APGEN?)

2. Poses a wider variety of questions on the same behavior model
   - For DR scenario, simulation and scheduling:
     • **What-if**: What events will occur and what happens to load and generation if responses are intercepted?
     • **When**: When do each of the events of the DR process take place?
   - Also capable of planning and model checking:
     • **What to do**: What events must execute to satisfy the constraints/achieve goals?
     • **Is possible**: Is it possible for an execution to cause a failure?
     • **Is impossible**: Is it impossible for an execution to cause a failure?

3. Scales well for problem size, ignoring problem complexity:
   - 10K+ events
   - 10K+ state variables (timelines)
   - 300K+ constraints
   - similar to ASPEN
   - CPLEX > million constraints—we plan to integrate solvers like CPLEX

4. Integrates with multi-view modeling (SysML) with access to ontologies.
Why is an expressive language important?

We need an expressive behavior modeling language for modeling information flow, timing, uncertainty, etc.,

Specifying meters for a scenario alternatives:

**BAD, but it’s the current state-of-the-art**

```c
operator ami_meter_1_sends_last_gasp_message (  
    pre: meter_1_Last_gasp_message = false  
    post: meter_1_last_gasp_message = true  
)  
operator ami_meter_2_sends_last_gasp_message (  
    pre: meter_2_Last_gasp_message = false  
    post: meter_2_last_gasp_message = true  
)  
...  
operator ami_meter_50000_sends_last_gasp_message (  
    pre: meter_50000_Last_gasp_message = false  
    post: meter_50000_last_gasp_message = true  
)  
```

**GOOD**

```c
class AmiMeter inherits from MeshNetworkTransceiver {  
    Messages messages = { LAST_GASP, READING, . . . }  
    ...  
}  
AmiMeter meters[50000];
```
AUI: Posing Analysis Questions

What **Load Reduction** would cause **grid.instability = true**? 117MW

What **Load Reduction** could cause **grid.instability = true**? [9.2MW, infinity]

If **Load Reduction = 15MW**, what **DR Area** could cause **grid.instability = true**? {7,13,51}

Load execution data for past 12 hours... done

Query: What unexpected events caused **grid.instability = true**? expected numDrParticipants = 4200 actualNumDrParticipants = [0,12]
How would you pose alternative questions if using other systems?

- ASPEN – figure out how to change activity and/or state/resource models to ask question
- CPLEX – figure out how to change model as a system of equations and an optimization function
- Simulink – change model and write MATLAB code
- Mathematica – figure out Mathematica code
- Wolfram SystemModeler – edit model (either graphically or in Modelica) and pose question in Mathematica.

In our approach,
- For the AUI, a new query statement template is added to others in a text file or GUI form:
  ```
  query HowMany:
  parameters = // format: [<type> variable|expression <parameter name>]*
  Number variable numVarParam1
  Boolean expression boolExprParam2
  statement = “How many ” + numVarParam1 + “ could cause ” + boolExprParam2 + “?”
  statement = “For how many ” + numVarParam1 + “ is ” + boolExprParam2 + “ possible?”
  ```
- In AE, add code (2 lines for this example) to an auto-generated Java class to change the model (in memory, not the original).
  ```
  code on next slide
  ```
- Now, this “HowMany” question can be asked of any model, for any variable in the model, and for any expression involving those variables.
AE event/behavior/constraint language

- Adds declarative behavior language elements to procedural Java for problem solving.

- **Classes** (OO inheritance, nested classes, leveraging Java)

- **Parameters**, a.k.a. variables with value domains

- **TimeVarying** – a.k.a. timelines, variables whose values are functions of time

- **Dependencies** (e.g., energy <- power * duration)

- **Constraints** (e.g., event1.end + 5 min < event2.start)

- **Events** – classes with start/end time variables
  - **Effects** – dependencies on TimeVarying
  - **Elaborations** – a.k.a. conditional decompositions, AND/OR event trees, subactivities, subgoals, methods, hierarchical task networks...

# This example is not an actual model.
# The syntax is modified to fit the screen.

class Customer:
    Parameter int id
    Parameter CustomerType type = Residence
    Parameter bool participate = false
    Parameter Meter meter
    TimeVarying float load = new TimeVarying("kv"+id)
    Dependency id <- meter.id

event usePower:
    Parameter float power, actualPower
    Parameter DRObj dObj
    Parameter time lastReport, nextReport
    Parameter bool willReport
    Parameter string fileName
    TimeVarying float projectedLoad
    Dependencies
energy <- actualPower * duration
    actualPower <- power - if(participate,0,
                               dObj.shed(power,id))
    fileName <- dataFolder + os.sep + "meter" +
                  id + "_" + startTime.day() + ".csv"
    nextReport <- lastReport + dObj.reportPeriod
    willReport <- endTime < new TimeVarying(fileName)
    projectedLoad <- new TimeVarying(fileName)

Effect
    load.add(power, startTime, endTime)

Elaboration
    if participate meter.report(load=load)

Constraints
    !participate || !willReport ||
    ( report.startTime >= nextReport - 2min 
      && report.startTime <= nextReport + 2min )
**TimeVarying (Timelines)**

- **TimeVaryingMap<T>**
  - setValue(time, value)
  - unsetValue(time, value)
  - plus(number, start, end)
  - plus(TimeVarying)
  - minus, times, dividedBy
  - init(csvFileName)

- **LinearTimeline<Number>**
  - initFrom(deltaMap)
  - getDeltaMap()

- **TimeVaryingMaps<T>**
  - init(folderName)
  - init(map<csvFileName, weight>)

- **Consumable<Number>**
  - plus(number, time)
  - getDelta(t1, t2)
  - getDeltaMap()

- **TimeVaryingList<T>**
  - add(time, value)
  - add(time, List)
  - addIfNotContained(time, value)
  - remove(time, value/List)
  - contains(time, value)
  - nthElement(time, n)

- **ObjectFlow<T>**
  - send(time, value)
  - sendIf(time, value, condition)
  - receive(time, value)
  - gotSomething(time)
  - addListener(ObjectFlow)
Constraint Solver

1. gather constraints
2. assign new values
3. elaborate or deconstruct events
4. apply (on unapply) effects to timelines
5. repeat
Logged output

- stats after each loop through constraints
- all constraints
- violated constraints
- execution/solution (events, parameter values, timelines)
- simulation – print event start/end and state transitions in scaled time
- snapshot simulations saved periodically during solving
Dynamic Plotter

• Enthought Python
  – doesn’t integrate well with Jython (and, thus, MD)
  – invoked as standalone from file or over a socket from Java (and probably Jython).

• options for scrolling, dynamic resizing, frames per second, skipping frames to catch up with simulation, saving movie (mp4)

• does not (yet) simulate by itself, so loads from log files are not animated

• supports projected and “live” data
  – can update projections

• currently some discrepancies in rendering from Java vs file because of sampling and handling of null values

<show plot animation>
Activity Diagram Animator

- time-scaled simulators in Java and Python can drive
- corrects for time error by monitoring system time
- data from log file or Solver
- max delay between event steps

show MD animation