Storm Framework

(Formerly known as Tosca)

Douglas Pearson
Nick Gorski, Rick Lewis, John Laird

Soar 27: May 24th, 2007

doug@threepenny.net

ThreePenny
S O F T W A R E

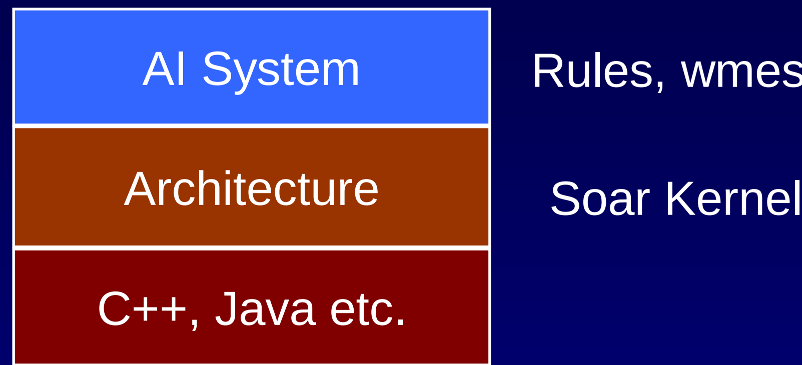
The Really Big Goal

- Research in Cognitive Architectures
  - Architecture: Cognitive systems built on top of a fixed set of capabilities
  - In some sense all Soar research contributes to this goal
  - Exploring the space of possible architectures
  - A major focus of John's career

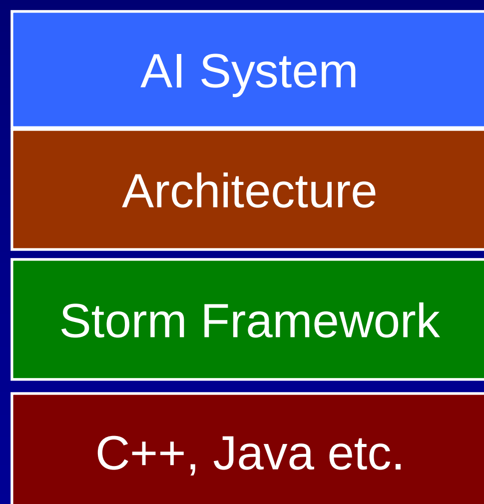
- But
  - Research at the architectural level is difficult
  - Theoretical concepts are easily muddled with implementation details
  - Architectural components are hard to replace or experiment with
    - E.g. Changing Soar's working memory implementation would be v. hard
  - Hard to evaluate the contribution of different architectural components
  - Biological constraints (brain mapping) is in the eye of the researcher

The Proposal: A Framework

- Current approach:



- Proposal:



Architectures make it easier to do research on AI systems

Hope is the Storm framework makes it easier to do research on architectures

ThreePenny SOFTWARE

Major Goals for Storm Framework

1. Easy to experiment on architectural elements
   - Plug-in modules, quickly replaceable
   - Supports experimentation
   - Also supports distributed development well

2. Support heterogeneous collection of components
   - Allow researchers to work in a language they find convenient (C++, Java, R, Matlab etc.)
   - Each component executing in parallel on possibly different timescales
   - All integrated together into a single executing model by the framework

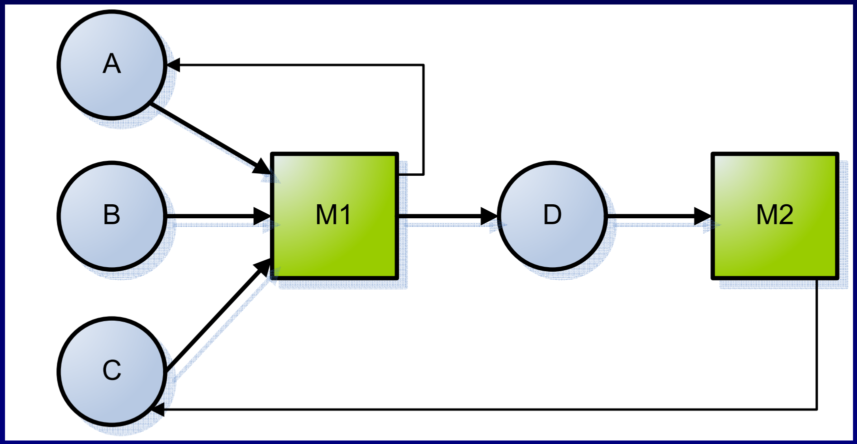
3. Ensure strong biological commitments made by architecture
   - Explicit representation of how processing maps to the brain
   - Explicit representation of which communication pathways are used in the brain

4. Flexible runtime configurations
   - Running on multiple operating systems
   - Running a simulation on a single machine
   - Running a simulation on a cluster of machines etc.

5. Flexible computational model
   - General focus is on flexibility
   - Architecture will likely be a more tightly defined and constrained model than the framework, which is more of a general purpose toolkit

Major Storm Elements

1. Function modules
   – Perform processing
2. State Variables
   – Hold all persistent data structures
   – Used for communication between modules
   – Examples:
     • Vector of floating point values (e.g. weights)
     • Frame buffer of image data
     • Control variables



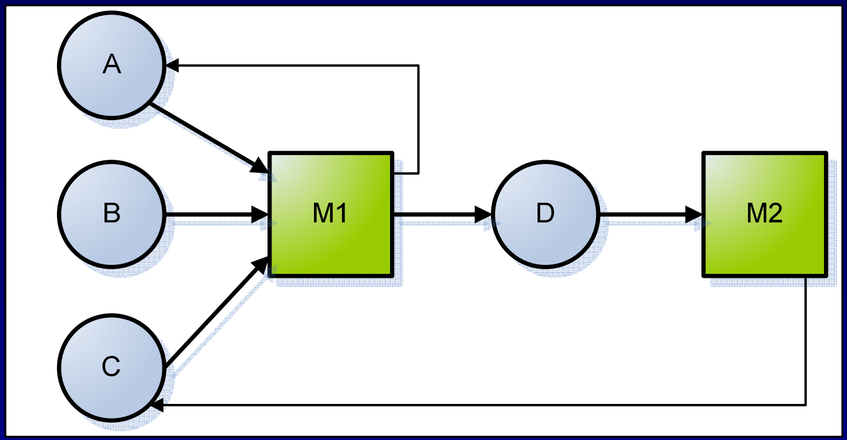
Define brain mapping and brain connectivity based on these elements, through 3 graphs:

- Functional connectivity graph
- Brain mapping graph
- Brain connectivity graph

ThreePenny SOFTWARE

Functional Connectivity Graph

- Establishes a mapping of state variables to inputs and outputs of function modules:

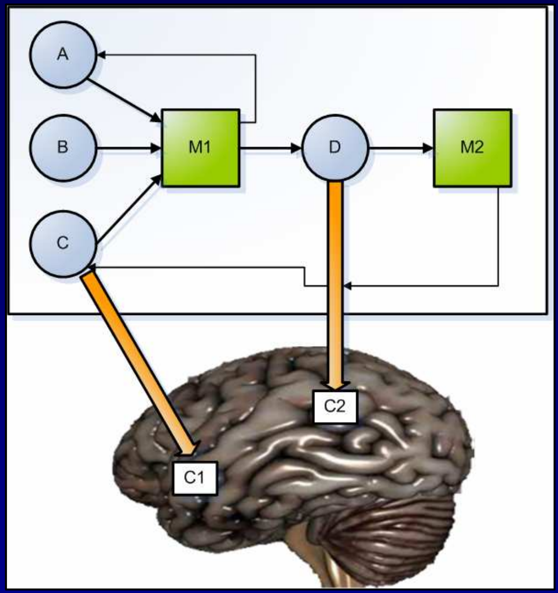


- Implicitly defined by associations between modules and variables
  - E.g. M1 registers inputs A,B,C and output D
- State variables used for all persistent data

ThreePenny SOFTWARE

Brain Mapping Graph

- Mapping from state variable to brain regions

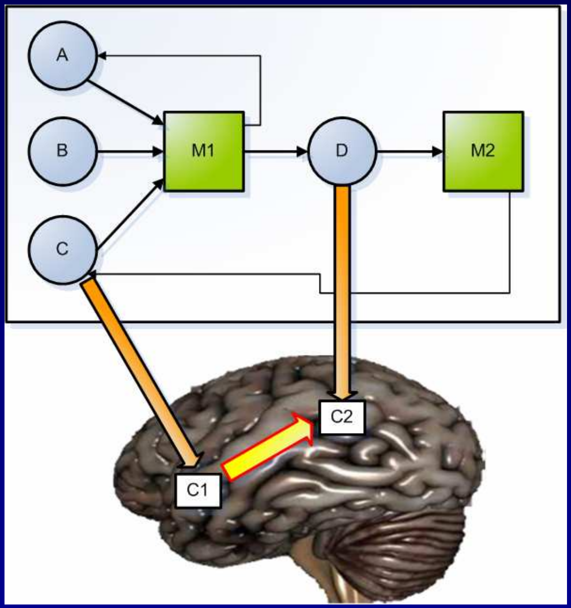


- Explicitly represented as a data structure in the framework
- Mapping will be complete
  - All state variables must map to a brain region
- May include unspecified regions
  - Makes gaps in the theory explicit

ThreePenny SOFTWARE

Brain Connectivity Graph

- Implied by function connectivity and brain mapping graphs



- Explicitly represent constraints on this mapping in the framework
- Automatic detection of constraint violations

ThreePenny SOFTWARE

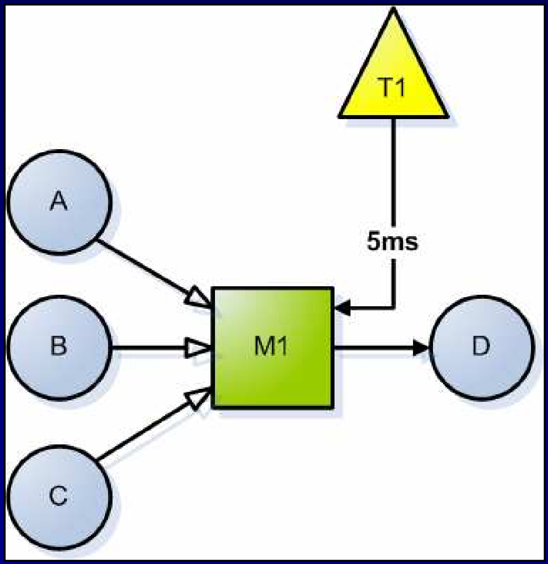
Sample Code

```
Value inputA = GetInput("A", time) ;
Value inputB = GetInput("B", time) ;
Value inputC = GetInput("C", time) ;

// Timescale need not be constant
ClockCannotAdvanceBeyond(time+5) ;

// Calculate the output

// Post the result
GetOuput("D")->SetValue(result, time+5) ;
```




```
GetInput("A")->NotifyWhenChanges(delay) ;
```

```
RegisterWakeup(time+10) ;
```

Event-driven : executes when input(s) change

Time-driven: executes at fixed rate

ThreePenny SOFTWARE

Framework Design

- **Framework provides an abstraction**
  - Each module is defined in terms of its own temporal constraints and with its own communication requirements
  - The framework handles the details of synchronization and handling actual communication

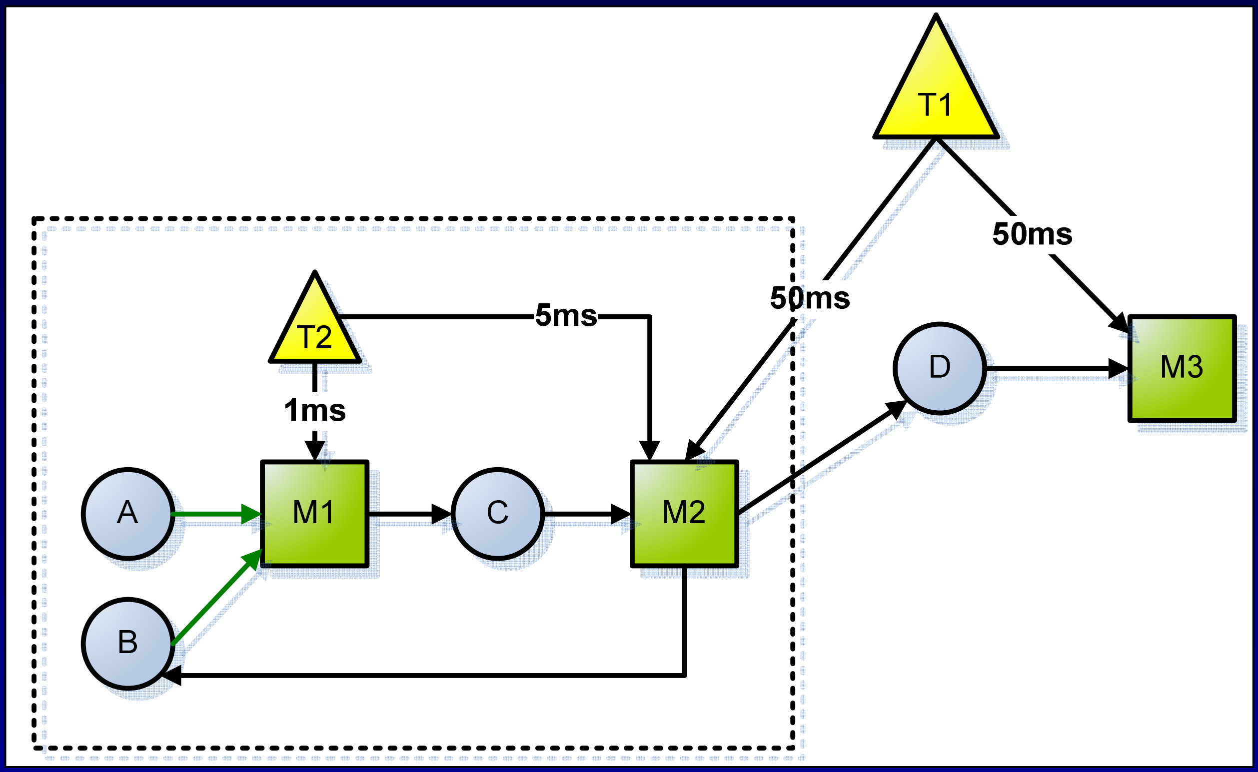
- Event triggers
  - *Clock-based* (after certain elapsed time) or
  - *Event-based* (after input changes)

  - *Delays :* inputs and outputs can include delays to model transmission time

- Flexible runtime configurations
  - Basing all comms on message passing between state variables, can generalize across:
    - Single process executing on one machine
    - Multiple processes using shared memory on one machine
    - Multiple machines communicating on a network
  - Current implementation executes each module in a separate thread
    - That base ensures separation between modules and explicit communication
    - Extends very well to multi-core revolution
    - Could support remote modules (machine clusters) in straightforward manner

- Language flexibility
  - Core implementation is in C++
  - Build interface modules to other languages using SWIG (same as used in SML).

Compose elements



Current implementation: Multiple communicating modules, different timescales.
No support yet for multiple clocks (should boost execution parallelism)

Summary

- Target is a flexible infrastructure layer that
  - Enhances experimentation
  - Forces brain commitments to be explicit
  - Scales flexibly to multiple machines, operating systems etc.
  - Supports a range of languages
  - Abstracts over communication details
  - Supports asynchronous execution
  - Supports a range of modeled time scales

ThreePenny SOFTWARE

Nuggets and Coal

- Nuggets – implemented features
  - Function modules
  - State variables
  - Hierarchical state variables
  - Connectivity graph
  - Brain mapping graph
  - Brain connectivity graph
  - Multi-threaded, single machine
  - Cross language (C++ and Java)
  - Tracing tool based on output logs

- Coal – not yet implemented
  - Multiple clocks / hierarchical function modules
  - Constraints violation detection
  - Cross machine communication
  - Linux, R and Matlab support
  - Higher level constructs for building modules/state variables
  - Proof that really supports experimentation