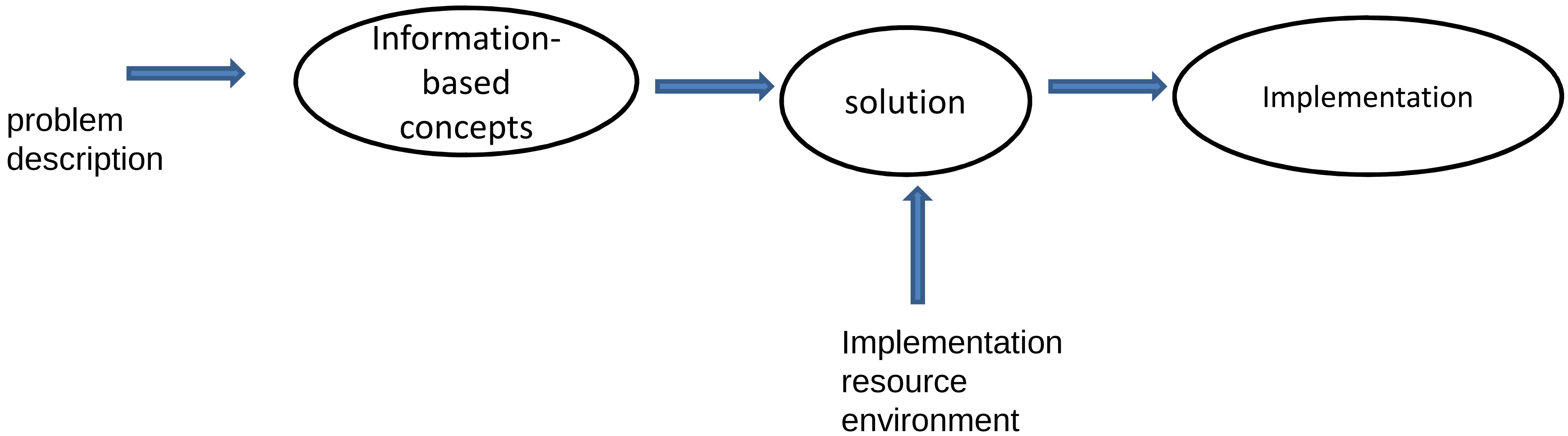
Activity tracking and awareness: A transdisciplinary automation framework

Alexander Muzy
Bernard P. Zeigler

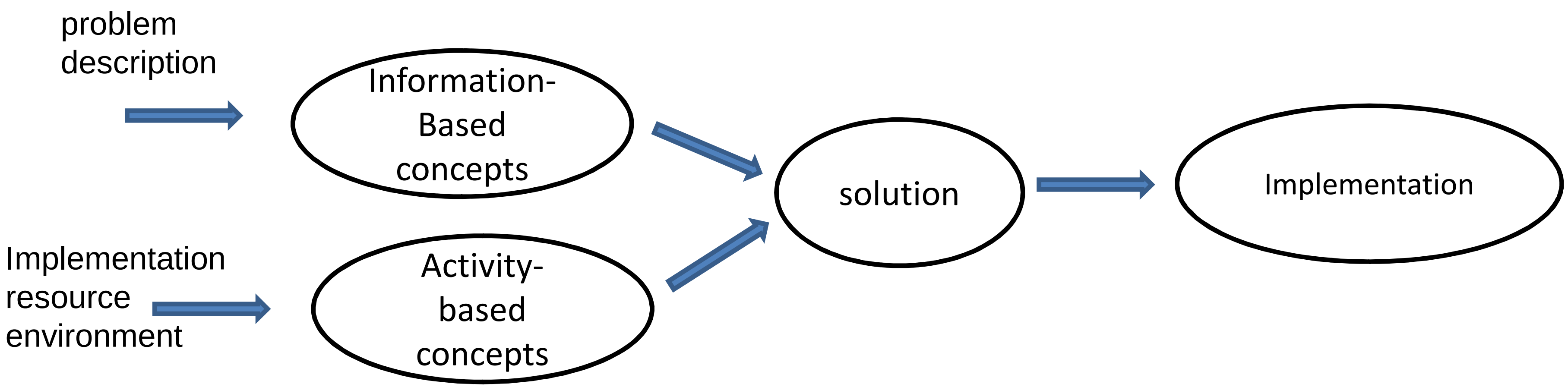
Activity Concept Hypothesis

- Activity is a generic concept (like “information”) refers to the spatial temporal distribution of state transitions in component-based model
- Activity concepts have been used to speed up simulation in the form of activity tracking which focuses computational resources on components based on their activities – it arises naturally in DEVS models with space/time heterogeneity (e.g. crowds, fires)
- Generalization Claim: Just as “information” is a useful abstraction for distinguishing behaviors from physical implementations, “activity” is a useful abstraction to enable energy consumption to be coupled to information flow for a more complete representation of how systems work
- Particular Hypothesis: “Activity awareness” can support “built-in” learning/adaptation similar to how it appears to work in biological systems, e.g. the brain

Today’s Information Technology



Tomorrow's Activity-Aware Information co-Technology??



Proposition – the implemented solution will be better because

- activity concepts allow a representation of the resource environment to be exploited earlier in the process
- the co-dependence of information and activity can be better understood, e.g., in how the brain constrained the development of mind
- activity measurement and exploitation can be built in to the implementation architecture to facilitate system development

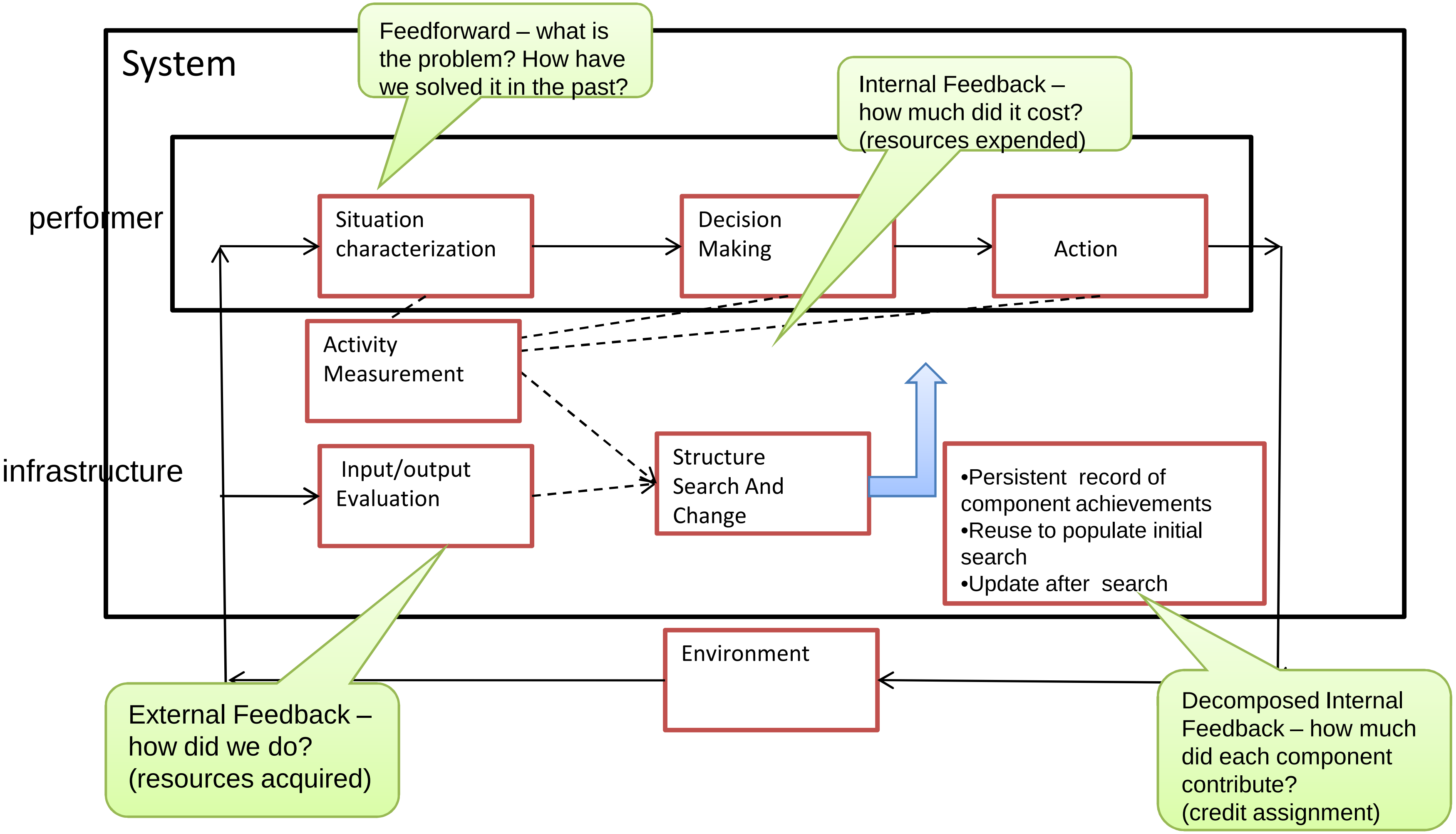
Biologically Inspired Activity-based learning/adaptation

- “Built-in” feedback for learning/adaptation requires credit to be apportioned to modules in proportion to their activity – naturally implemented as energy (bio-chemical resource) consumption supporting increased capacity to consume in the brain
- Fundamental hypothesis – modules that are highly active over the course of a successful trial are more likely to be responsible for that success than modules that are less (or in-) active in that trial.
- Activity-based learning/adaptation rule – high activity & success gets rewarded; high activity & failure gets punished (c.f. other rules, e.g., back propagation, bucket-brigade,…, that are not generic so are not “built-in”)

Activity-based learning/adaptation precursors in the literature

- Hebb's rule: neurons that are active concurrently have their synapse connections strengthened, co-active groups get more tightly connected
- Carruthers: Active modules can activate (start up) other modules in their "neighborhood", providing a structure exploration capability
- Spreading activation determines the nature of the search in solution space http://en.wikipedia.org/wiki/Spreading_activation,
- Minsky: agents (resources) that were active during a successful solution are remembered by a K-line and connected to the problem input description for later re-combination and re-use (recall Alexandre's formulation)

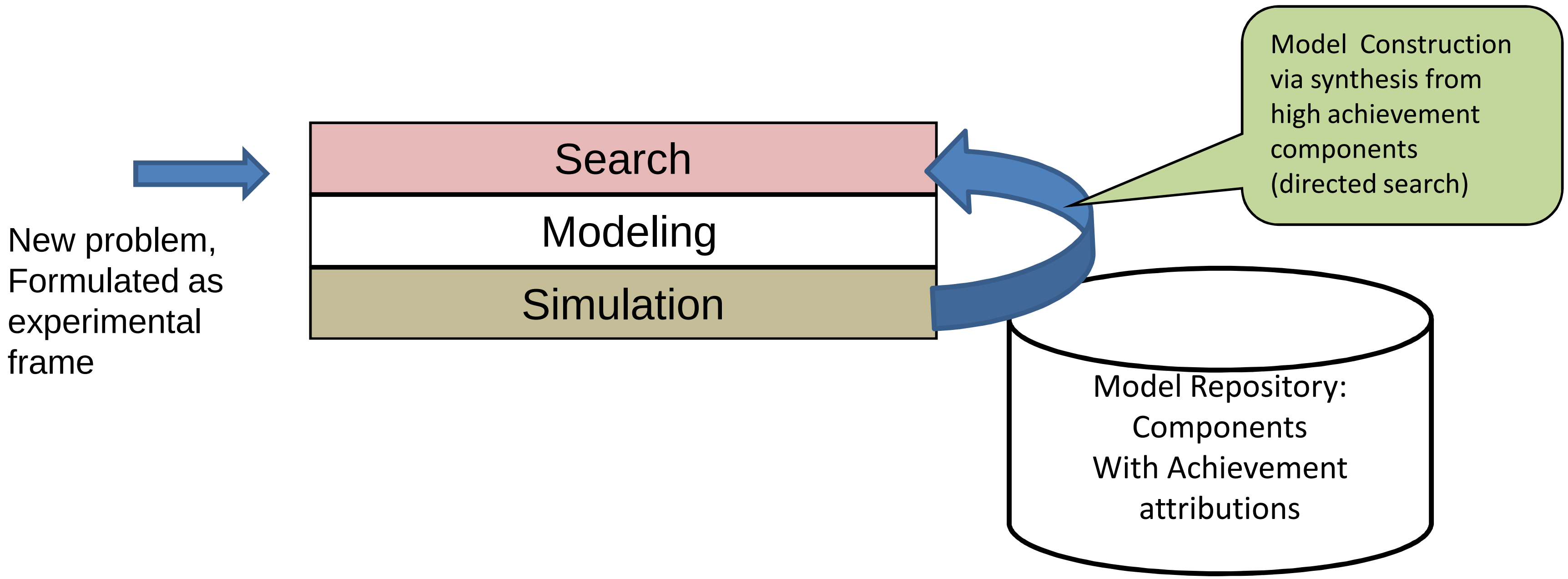
Activity-Aware System Architecture



Survive if resources acquired >= resources expended

Automating Model Construction with Built-in Learning and Component Re-use

New paradigm: Synthesis of model for a new objective is a search process which is accelerated by re-use of high achievement components



achievement determined by correlation of evaluation of, and activity participation, in previous outcomes

Analogy: building a better brain is like building a winning hockey team

feature	hockey team manifestation
collaboration requirement	team must work together, no player is sufficient
modularity	6 distinct positions on ice
specialization	each position has its own skill set
substitution alternatives	18 players on team, 6 on ice at any time, players get tired and are replaced Also farm club and trades furnish additional alternatives
problem	coach/manager must select 3 subsets of 6 that work best together to win games

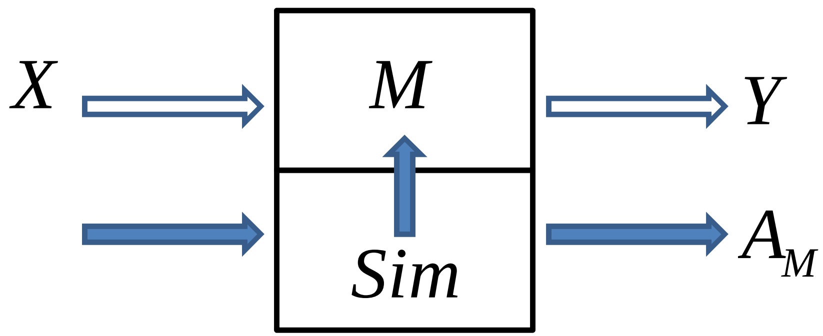
analogy mapping	players are reusable components, build team as a composition of players

feature	hockey team manifestation
trial	game = 60 minutes
activity of component	player's minutes on ice
evaluation of trial	game outcome, e.g. goals scored – goals allowed
credit assignment to component -correlation of activity and outcome	minutes played * evaluation of game
achievement stored in repository	accumulated credit over player past performance

How to Support Activity Awareness

M&S Infrastructure needed:	**DEVS capability**
components	atomic models
composition	coupled models
Support change in composition – also while simulating	Dynamic Structure
organization of models and management of substitutions	System Entity Structure
ability to collect activities and store in repository to support search	subject of this talk

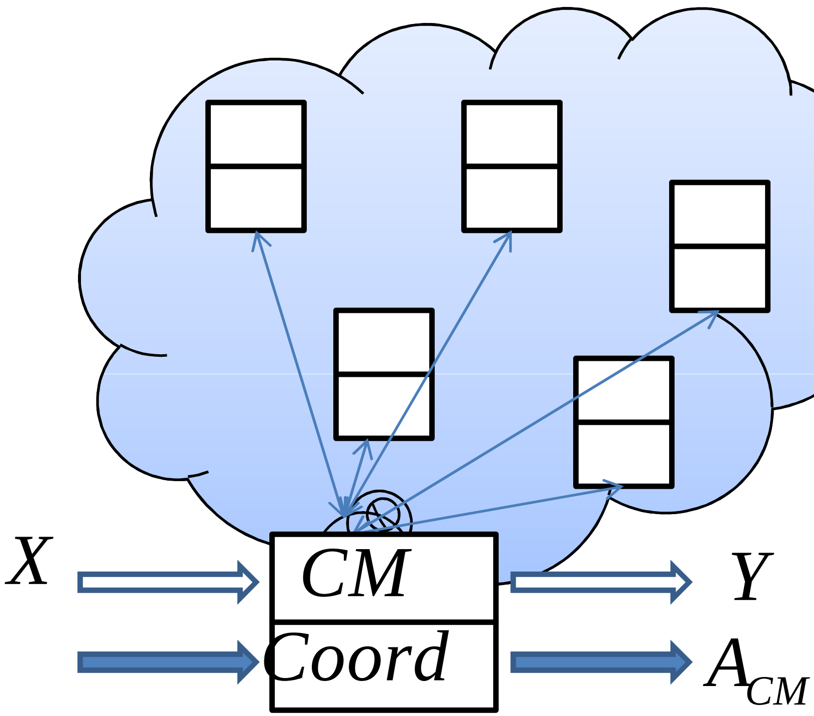
Activity Measurement in DEVS Atomic Model



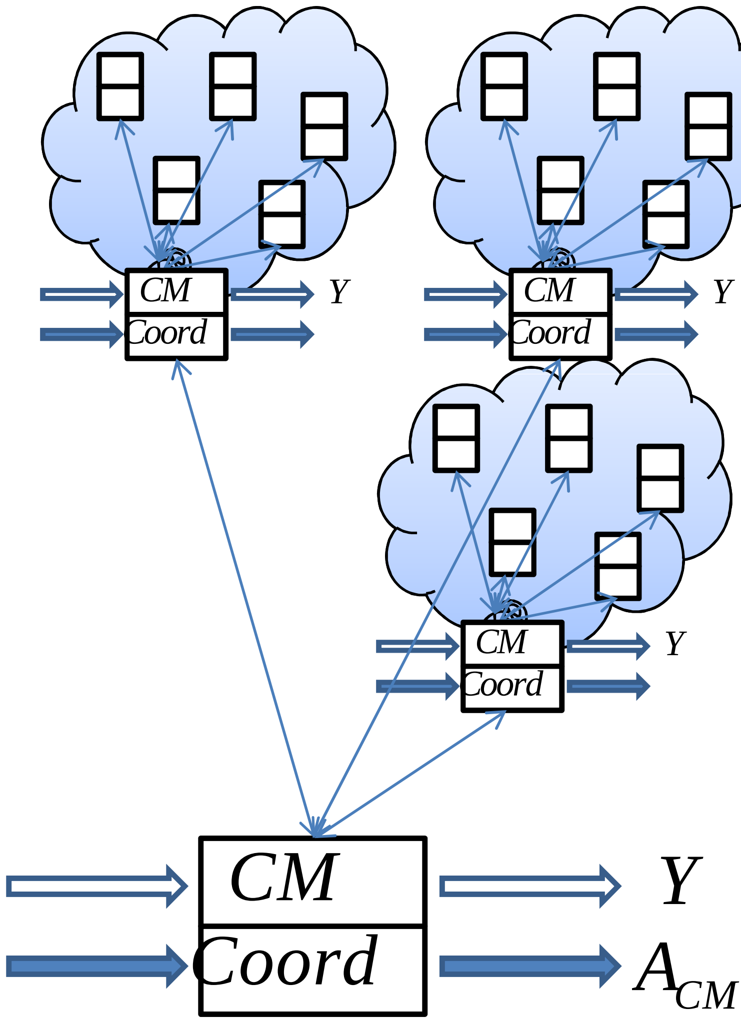
$$s' = \delta_{int}(s)$$
$$\Delta(s', s) > q \Rightarrow n_{int} = n_{int} + 1$$
$$A_{int}(t, t') = \frac{n_{int}}{t' - t}$$

$$s' = \delta_{ext}(s, x)$$
$$\Delta(s', s) > q \Rightarrow n_{ext} = n_{ext} + 1$$
$$A_{ext}(t, t') = \frac{n_{ext}}{t' - t}$$
$$A = A_{int} + A_{ext}$$

Activity Measurement in DEVS Coupled Model and Hierarchical Coupled Model



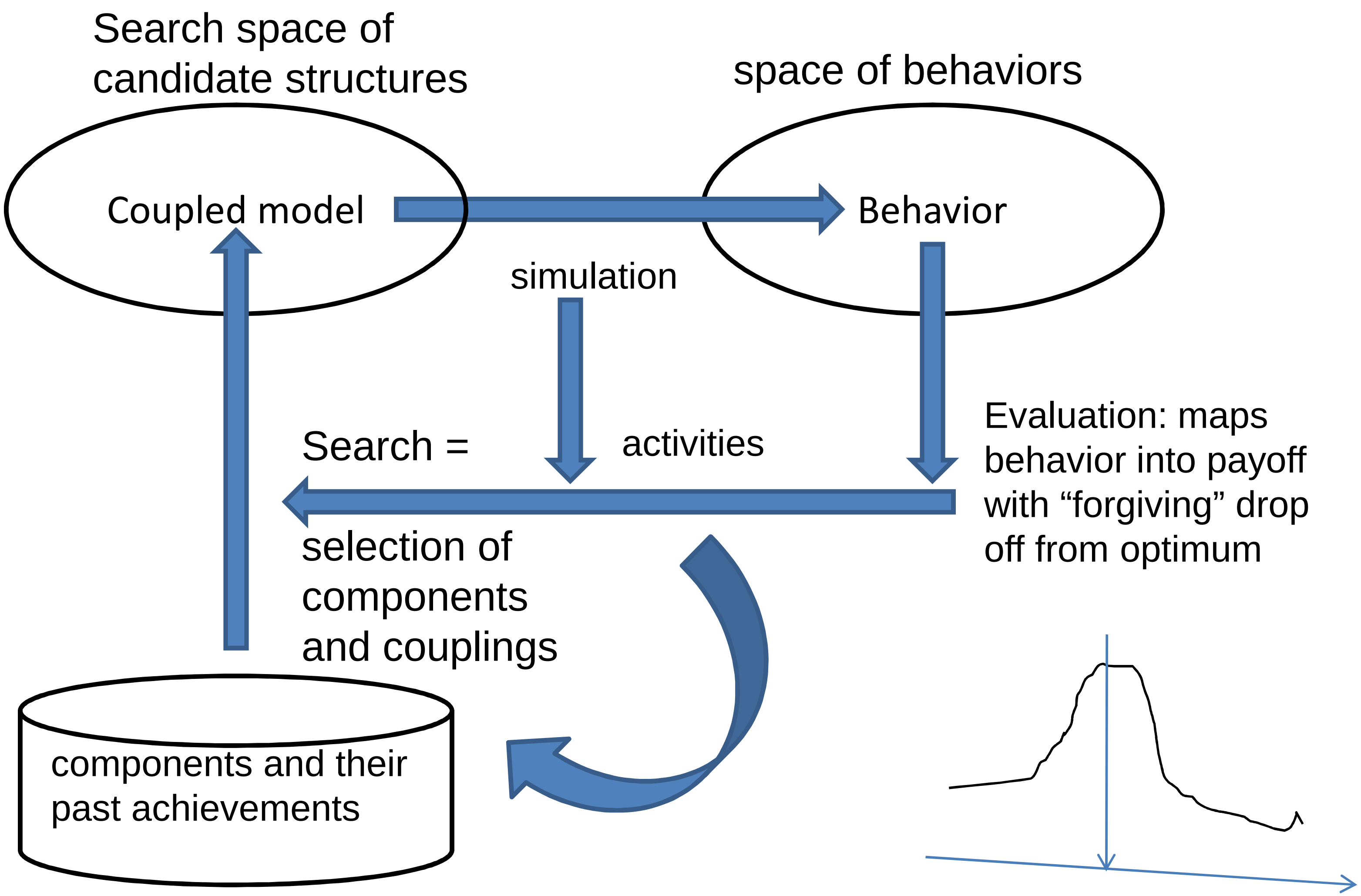
$$A_{CM} = \frac{1}{|D|}\sum_{M \in D} A_M$$



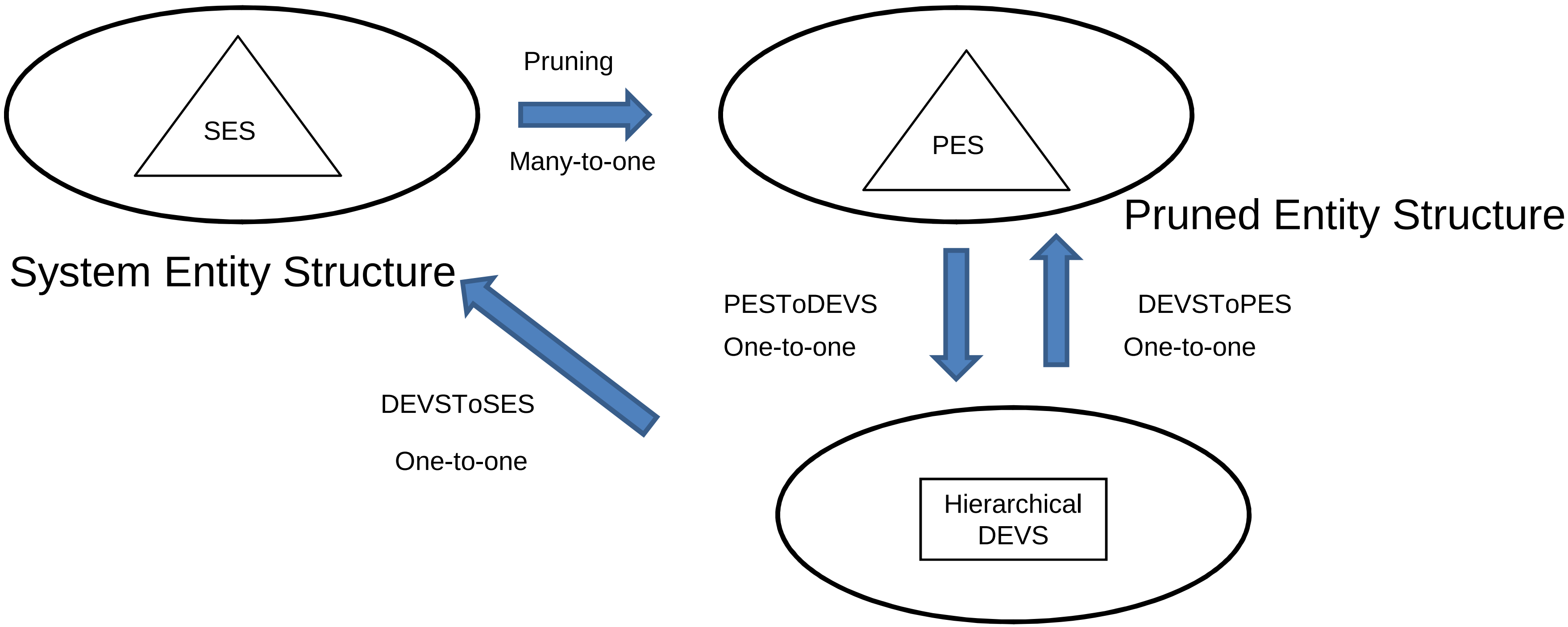
Aspects of Activity-Based Feedback

- Evaluation of output – score indicates quality, higher is better
- Total activity of candidate model- represents energy used, lower is better
- Individual component credit assignment – represents correlation of its activity with candidate scores over candidates in which it has participated
- For candidates with the same score, the one with lower total activity is better, e.g., can use score/totalActivity to compare (cf: benefit/cost ratio).
- This helps in search where current composition has redundant connections, then removing connection will not alter score but will reduce activity cost.

Overall Concept

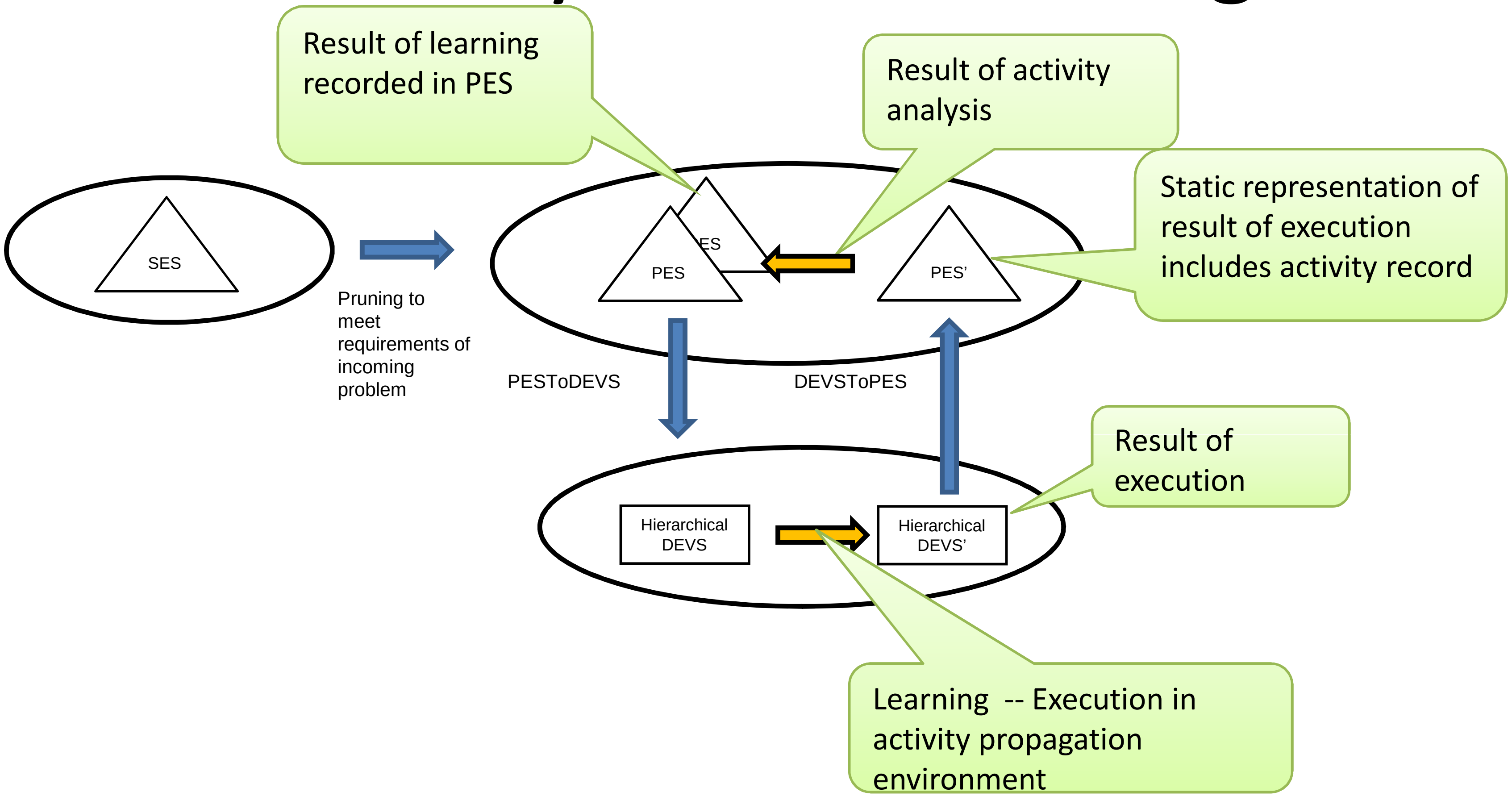


SES, PES, DEVS mappings

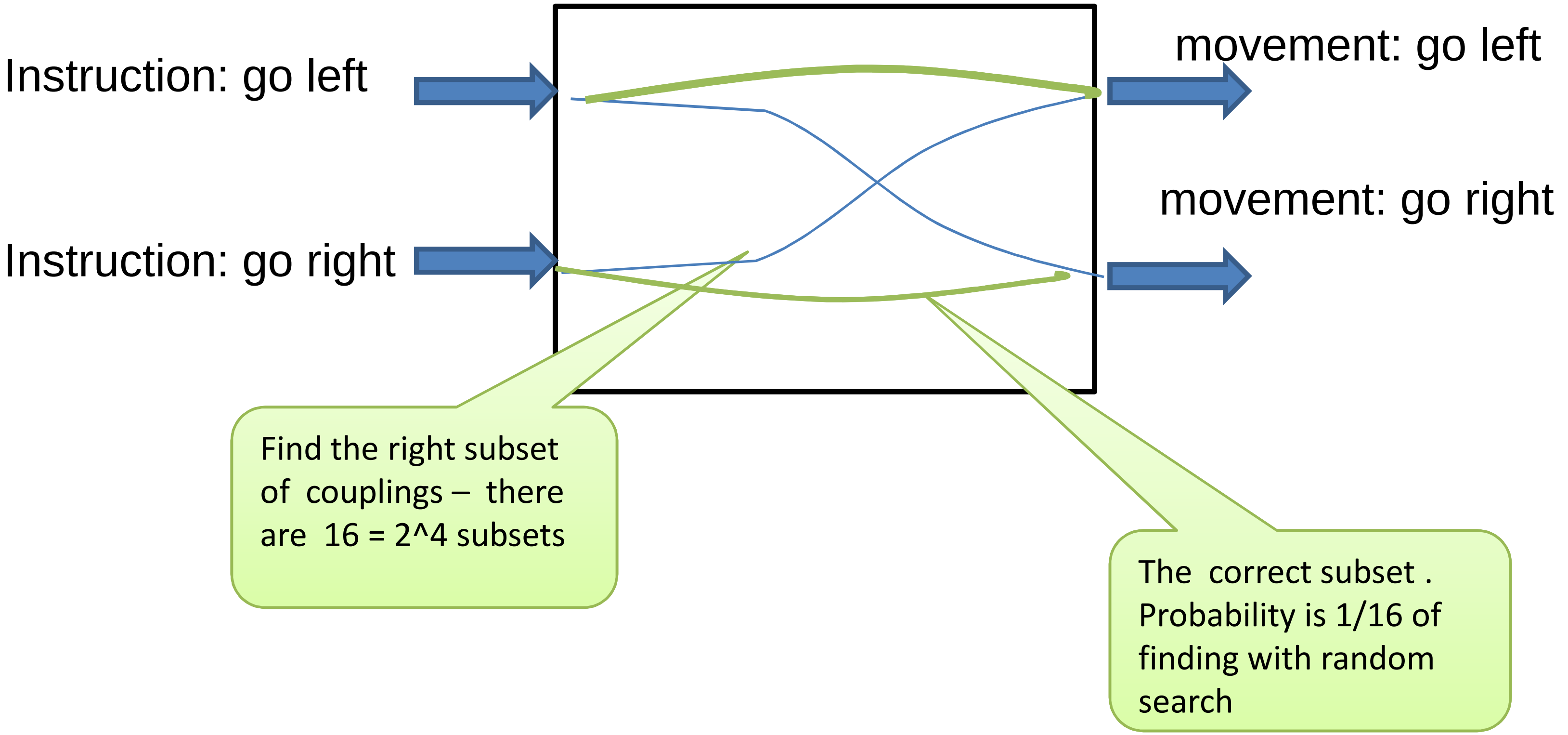


Since Pruning is many to one, DEVSToSES must arbitrarily select one SES that maps to the given DEVS

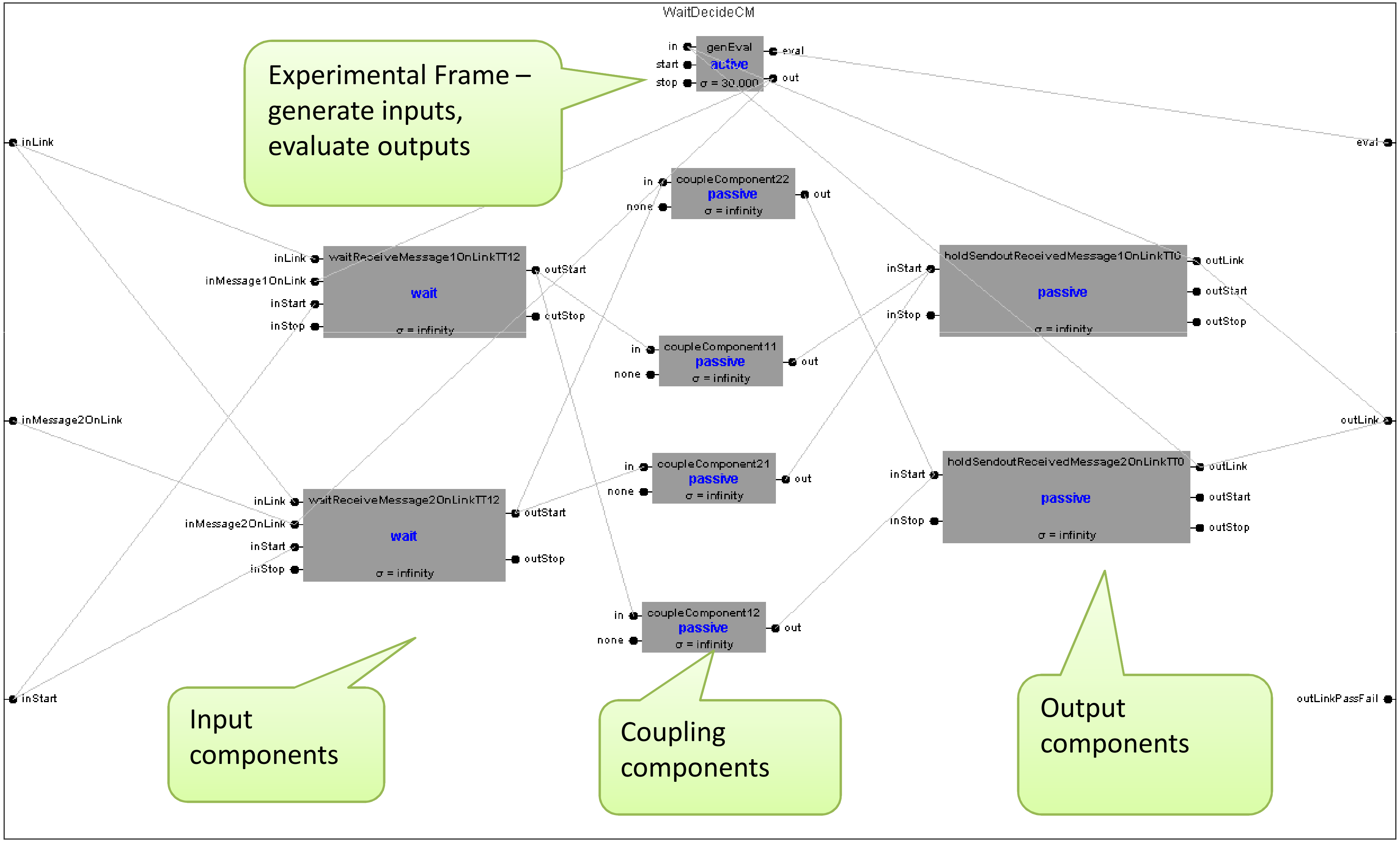
Activity Based Learning



Activity-based Learning Example



Activity-based Learning Example



Evaluation of output

Given $f : X \to Y$

Define $evaluate(f, x, S)$ with range in [0 ,1]

by

$$evaluate(f, x, S) = 1 \quad \text{if} \quad S = \{f(x)\}$$

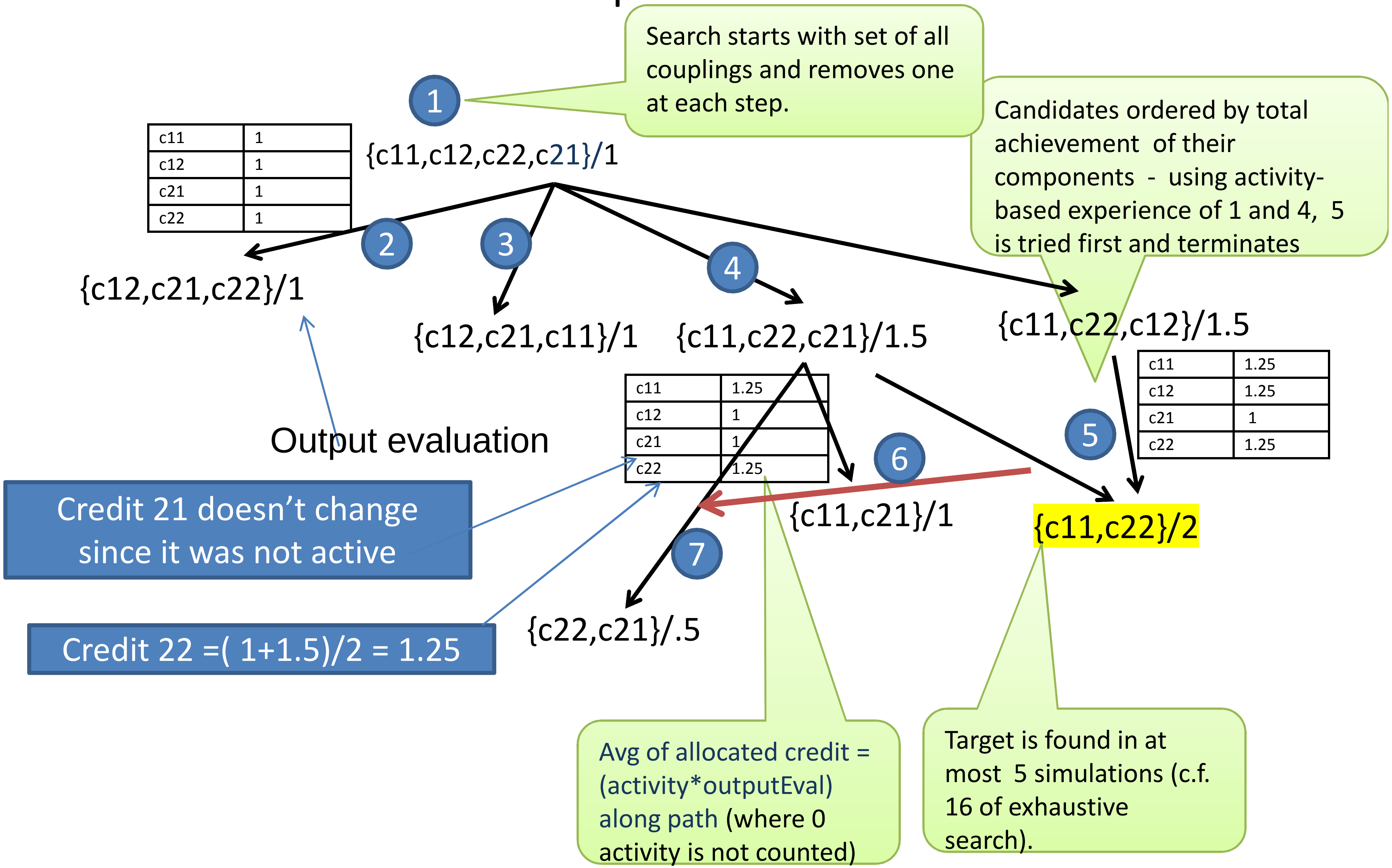
$$= \frac{val}{|S|} \quad \text{if} \quad f(x) \in S \quad \text{and} \, |S| > 1$$

$$= 0 \text{ otherwise}$$

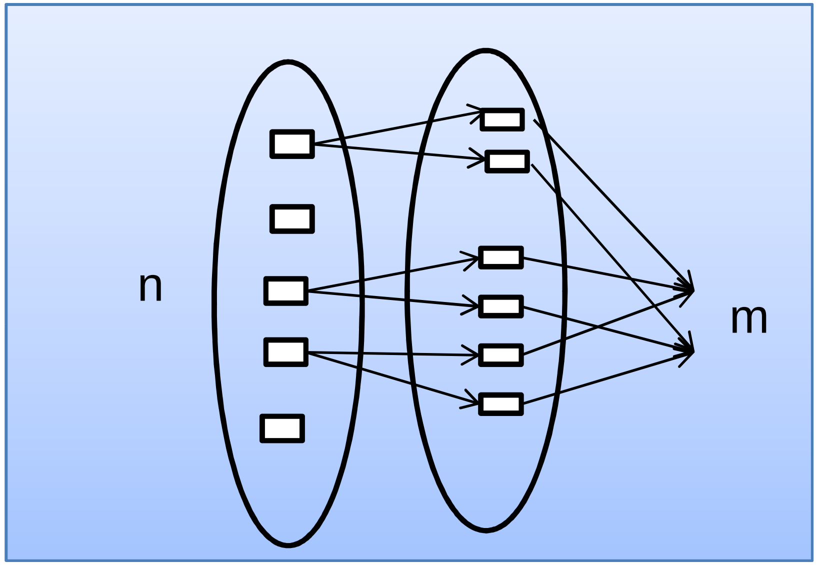
S is a subset of of Y. representing the outputs that were produced by the system when x was the input. The correct output is f(x)

Some credit for containing the right output based on a parameter, *val*, and decreasing as the number of other outputs increases.

Breadth-first Search – stop when score does not increase



Many-to-one Mapping



- N inputs , m outputs,
- the max score is n when every input is mapped to the correct output
- there are (n*m) couplings initially,
- requiring at most 2^(nm) evaluations required for exhaustive search.

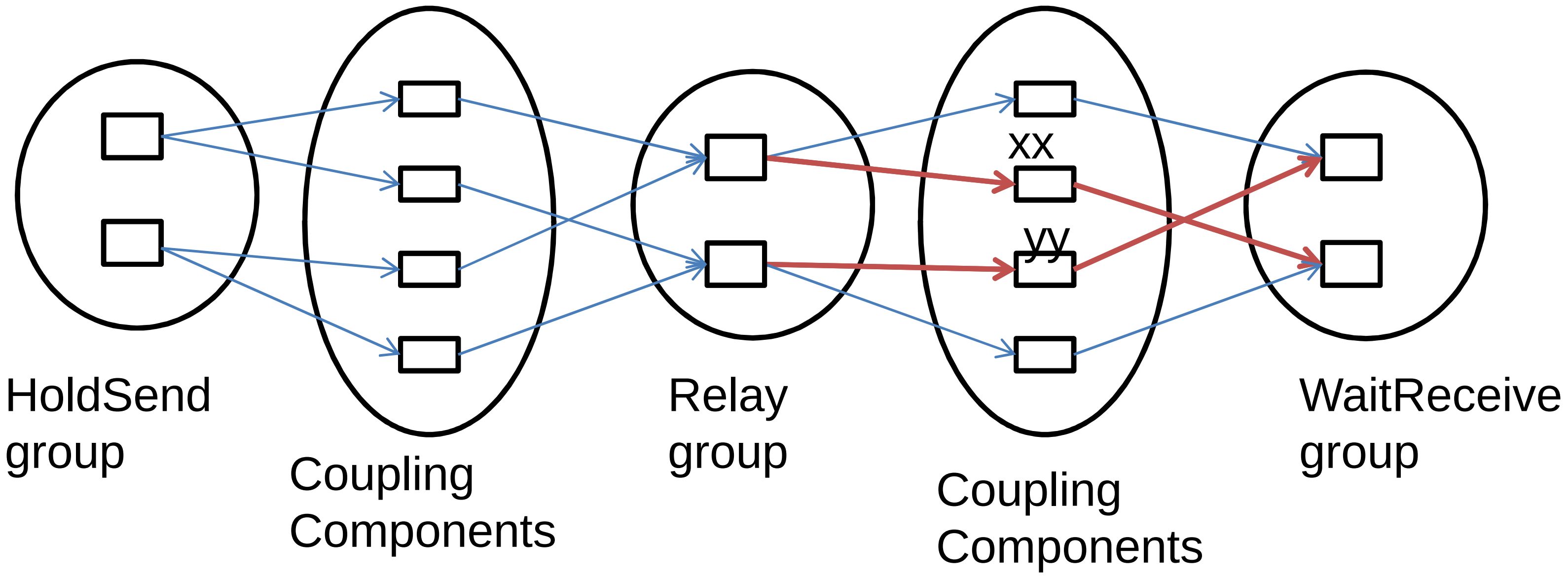
- start with the initial set of all couplings of size nm

At each stage, i,
- reduce the subset by one, i
- examine at most each of the (ni-1) subsets for the highest score at that stage
- stop when the right subset of size n is found

- Compare using component achievements vs with not using component achievements
- Can show that the hardest case is when n=m and for that the expected number of simulations is n^2 (with achievements) vs n^3 (without)

With achievement use , pre-order the sets by summing up the subset achievements

Harder



Number of alternative couplings = 16*16
Number of fully correct solutions = 2
Search space = 8*16 = 128

If remove xx and yy
Number of alternative couplings = 16*4
Number of fully correct solutions = 1
Search space = 4*16 = 64

If remove xx or any one coupling:
Number of alternative couplings = 16*8
Number of fully correct solutions = 1
Search space = 8*16 = 128

Experimental Results are consistent with these numbers

Interoperation vs Integration*

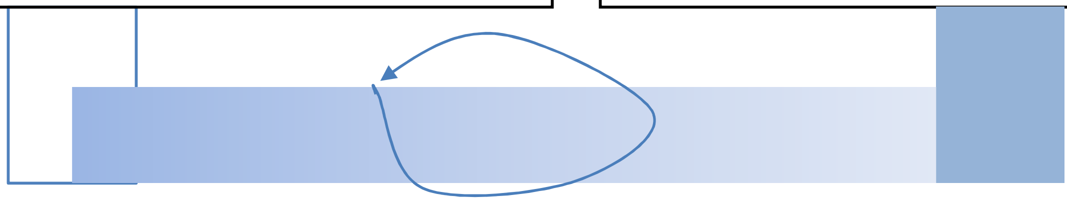
Interoperation of system components

- participants remain autonomous and independent
- loosely coupled
- interaction rules are soft coded
- local data vocabularies persist
- share information via mediation

Integration of system components

- participants are assimilated into whole, losing autonomy and independence
- tightly coupled
- interaction rules are hard coded
- global data vocabulary adopted
- share information conforming to strict standards

reusability
composability
System is adaptive

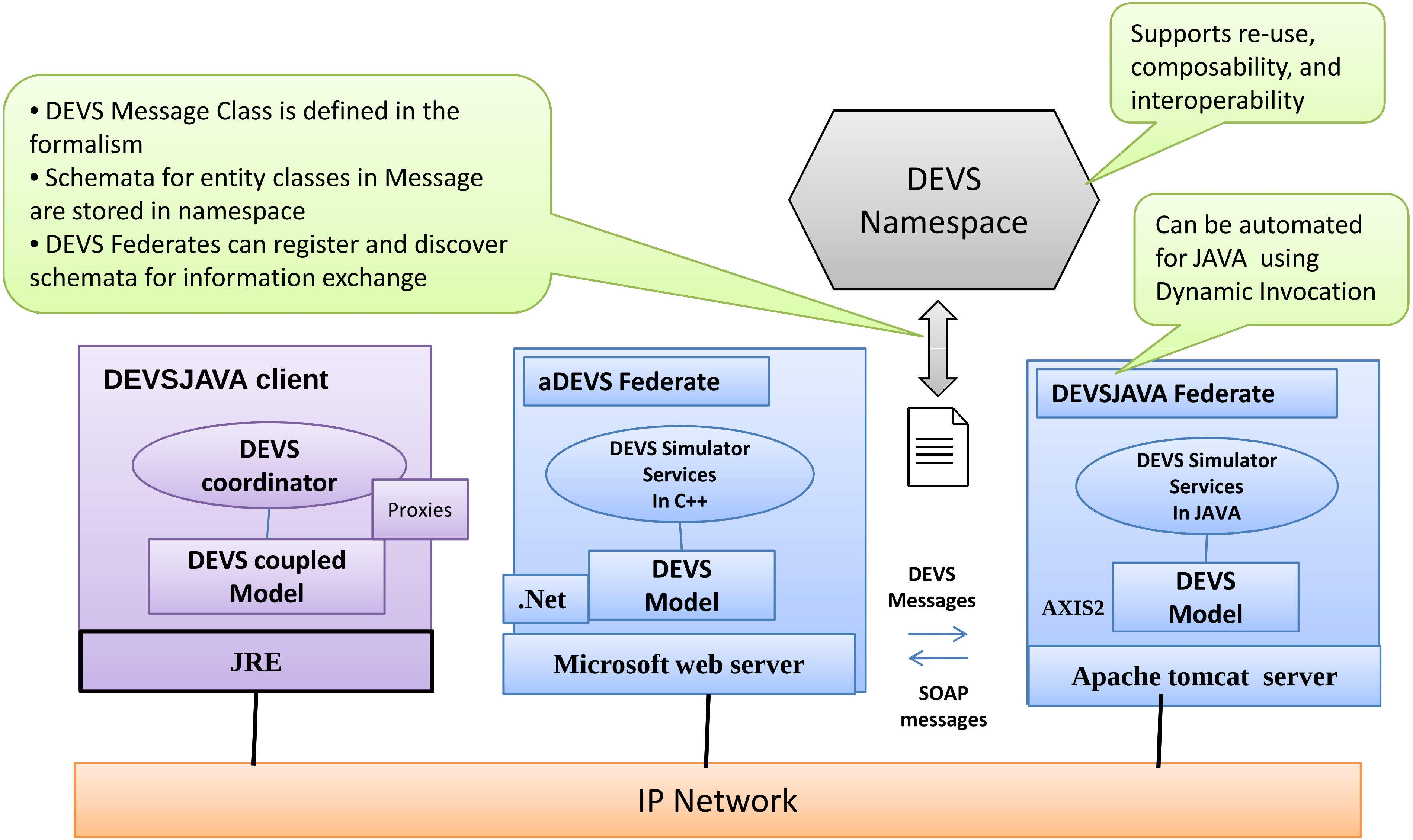


Efficiency
Non-adaptive

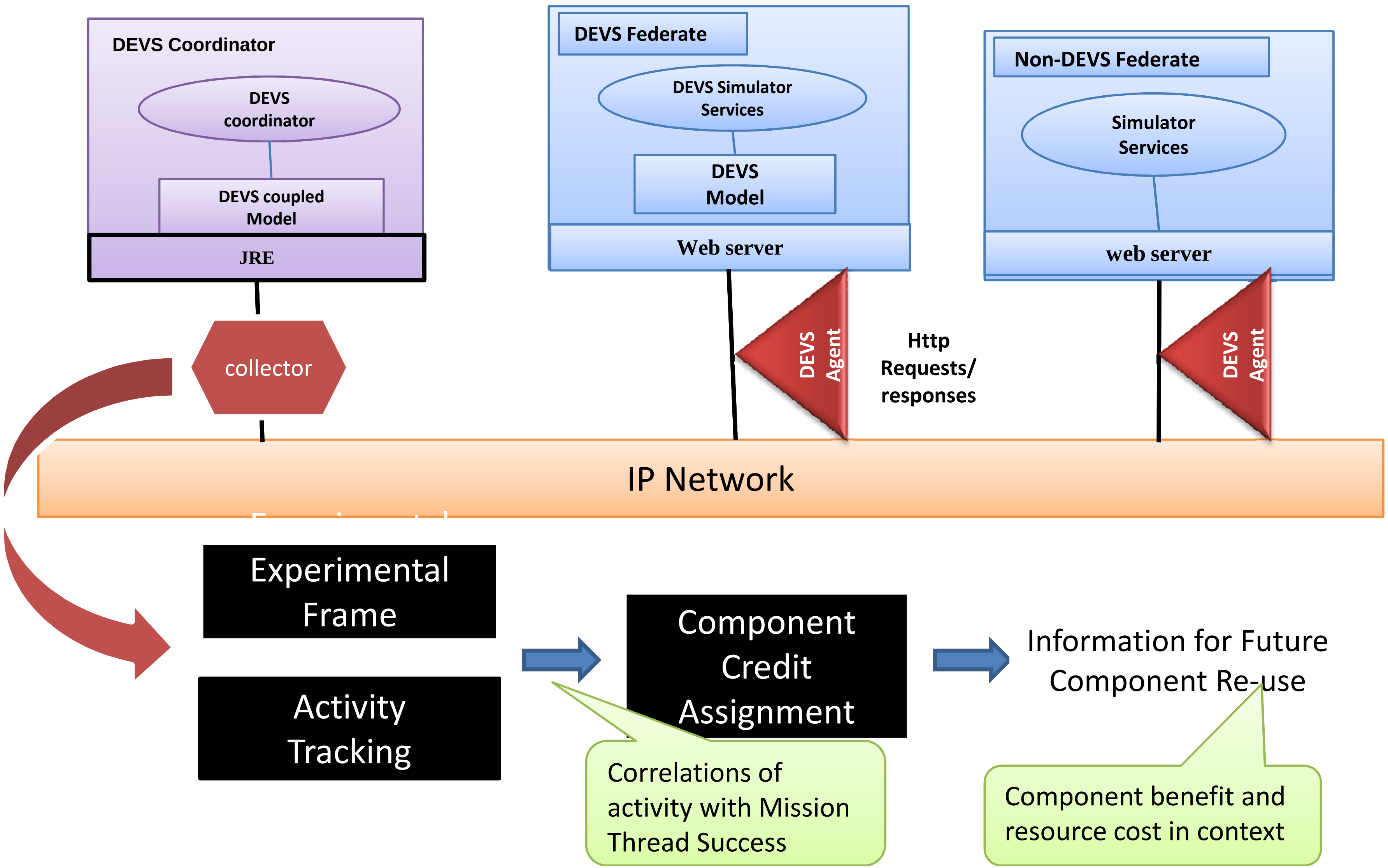
Edelman: fluctuate between these poles

* adapted from: J.T. Pollock, R. Hodgson, “Adaptive Information”, Wiley-Interscience, 2004

Web-enabled interoperability of DEVS components



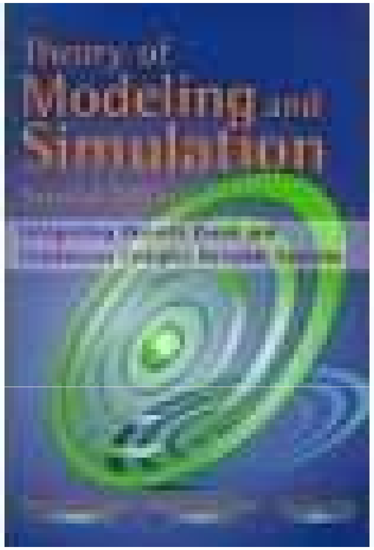
Activity-Based Evaluation for Web Component Re-use

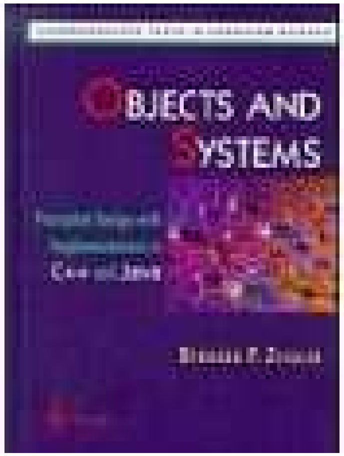


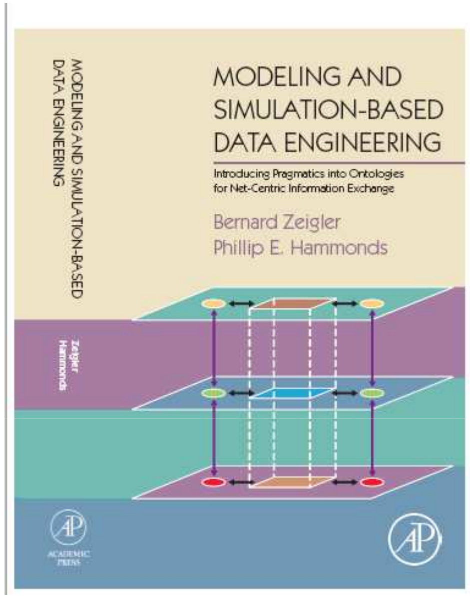
Some activity implications

- Activity tracking in crowd modeling and simulation (Xioalin)
- Activity tracking in graph transformations (Hans)
- Activity tracking of one agent of another (G. Deffuant)
- Activity awareness in theory creation (Levent)
- Activity inference patterns in component-based models (J.P. Briot)

Books and Web Links







devsworld.org

www.acims.arizona.edu

Rtsync.com

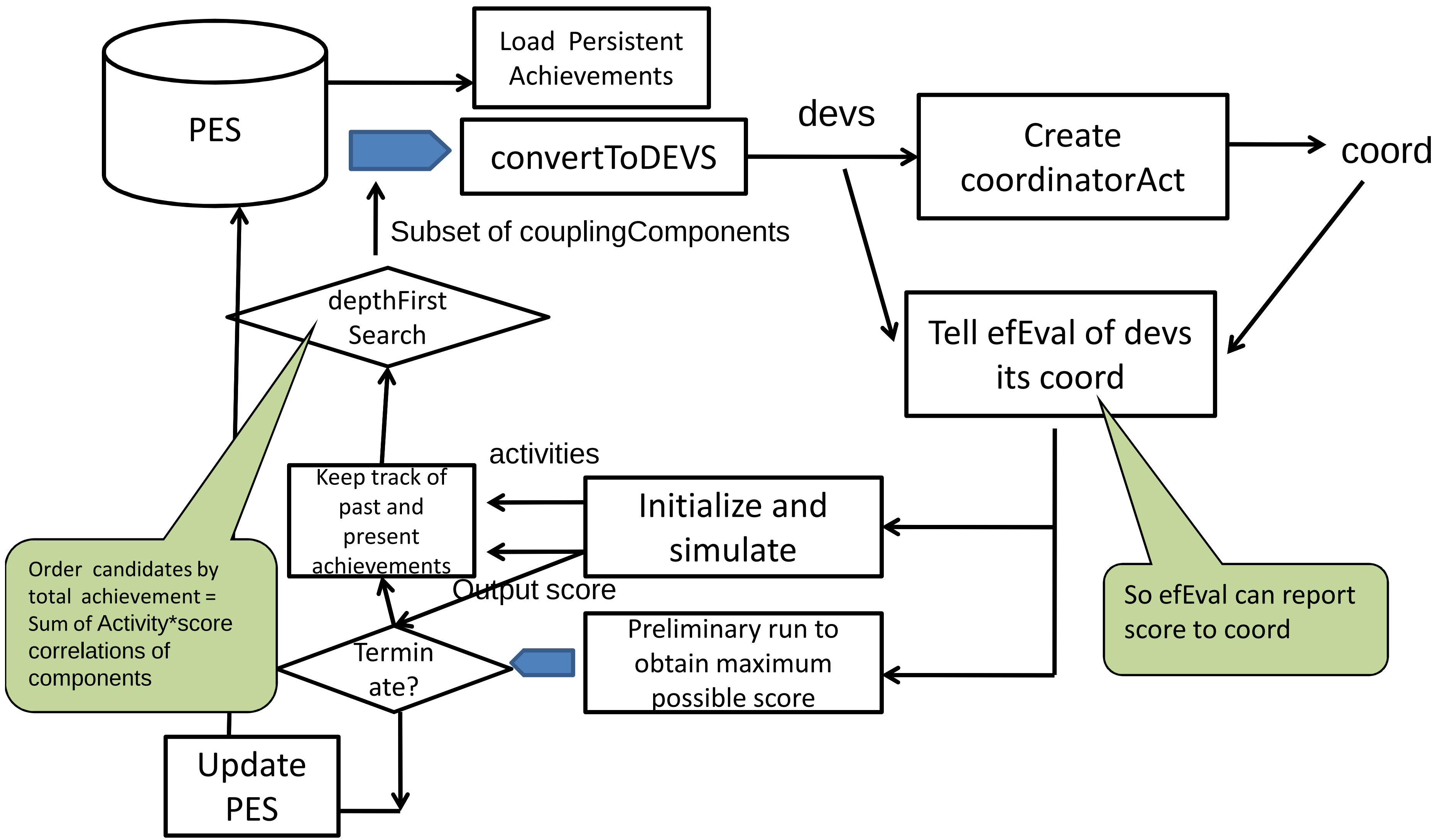
More Demos and Links

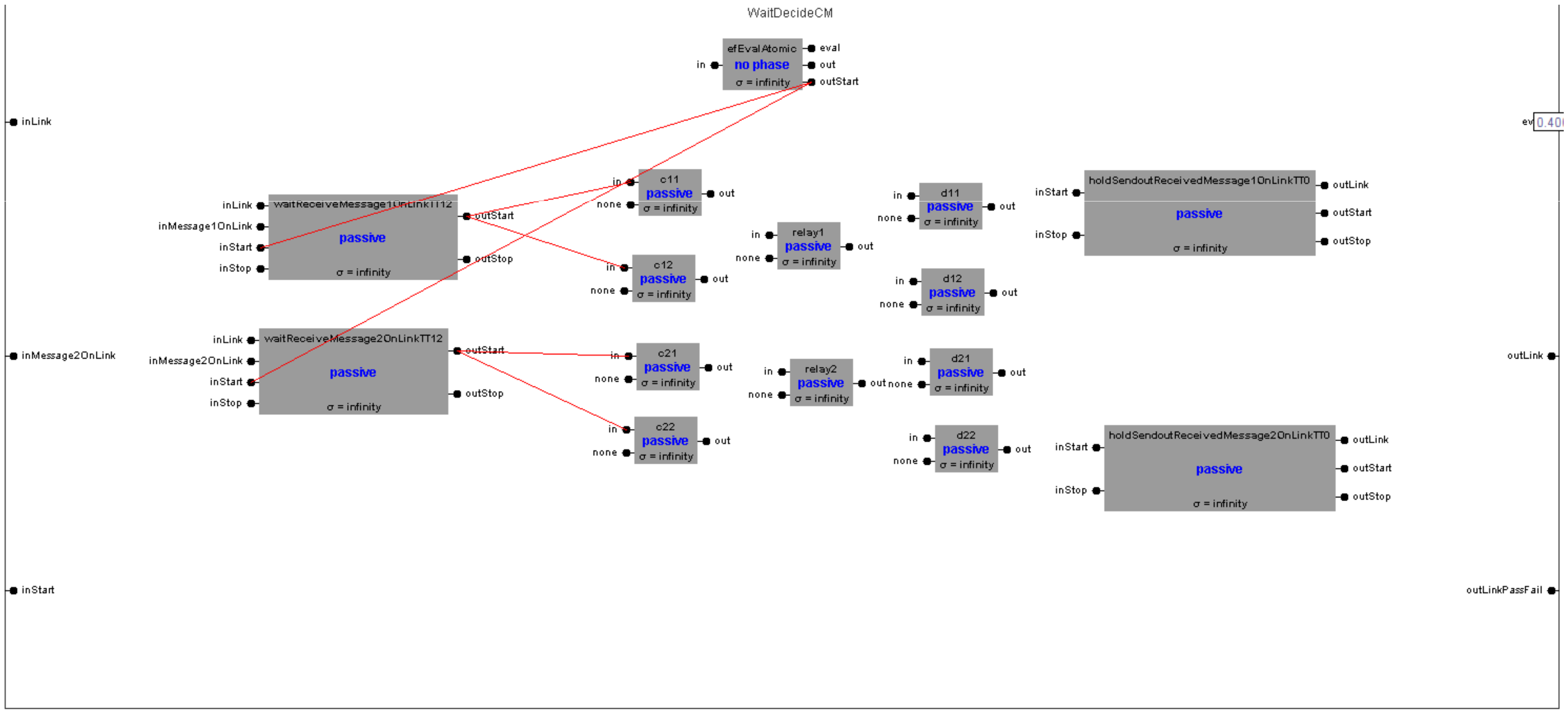
http://www.acims.arizona.edu/demos/demos.shtml

- **Integrated Development and Testing Methodology:**
- **AutoDEVS (ppt) & DEMO**
  - **Natural language-based Automated DEVS model generation**
  - **BPMN/BPEL-based Automated DEVS model generation**
  - **Net-centric SOA Execution of DEVS models**
  - **DEVS Unified Process for Integrated Development and Testing of SOA**
- **Intrusion Detection System on DEVS/SOA**

Backup

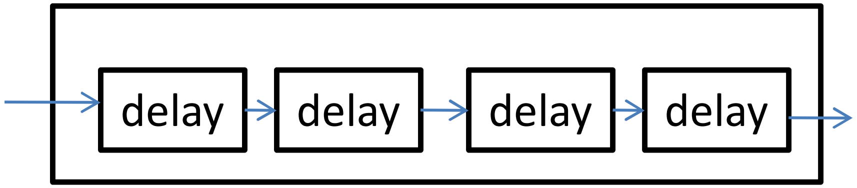
Search Algorithm Control of Simulation

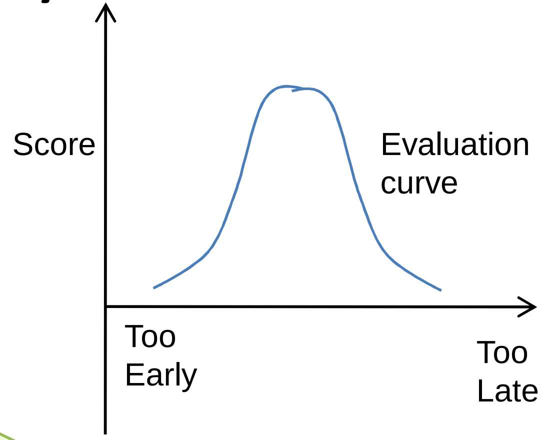




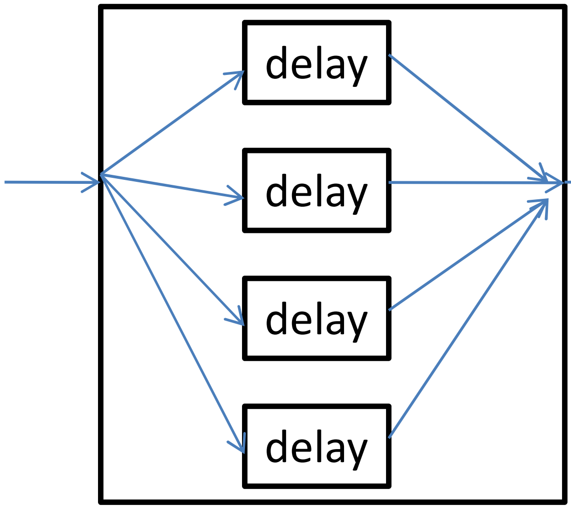
WaitDecideCM
efEvalAtomic
no phase
σ = infinity
in
eval
out
outStart
inLink
inMessage2OnLink
inStart
waitReceiveMessage1OnLinkTT12
inLink
inMessage1OnLink
inStart
inStop
passive
σ = infinity
outStart
outStop
waitReceiveMessage2OnLinkTT12
inLink
inMessage2OnLink
inStart
inStop
passive
σ = infinity
outStart
outStop
c11
c12
c21
c22
relay1
relay2
d11
d12
d21
d22
in
none
out
passive
σ = infinity
holdSendoutReceivedMessage1OnLinkTT0
holdSendoutReceivedMessage2OnLinkTT0
inStart
inStop
outLink
outStart
outStop
passive
σ = infinity
outLink
outLinkPassFail

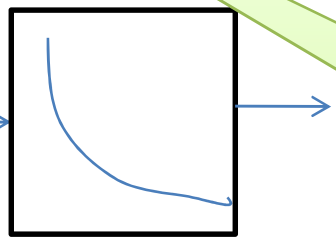
Series and Parallel Composition have opposite timing properties wrt activity based search





Credit to component = score/total activity

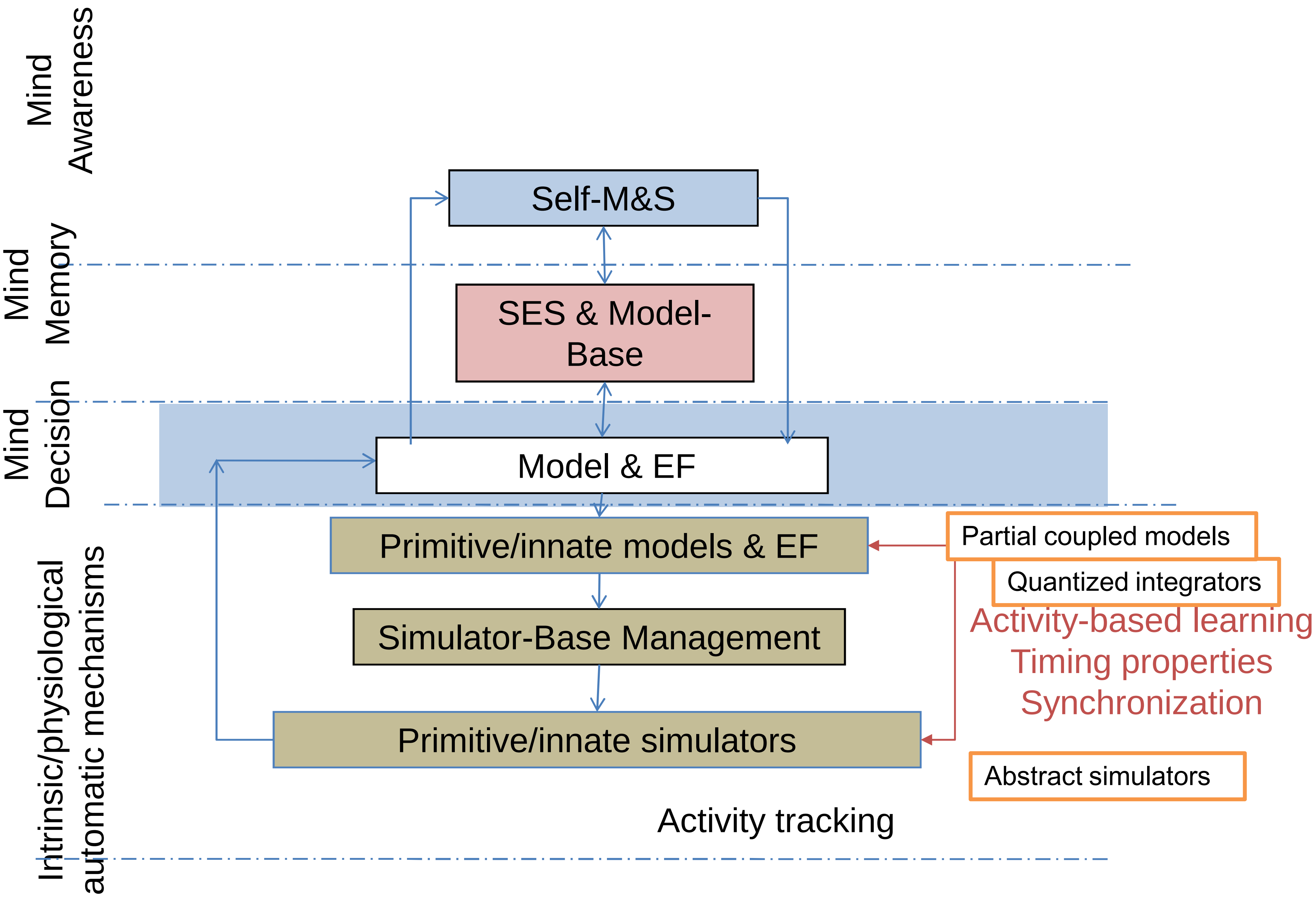




Threshold curve

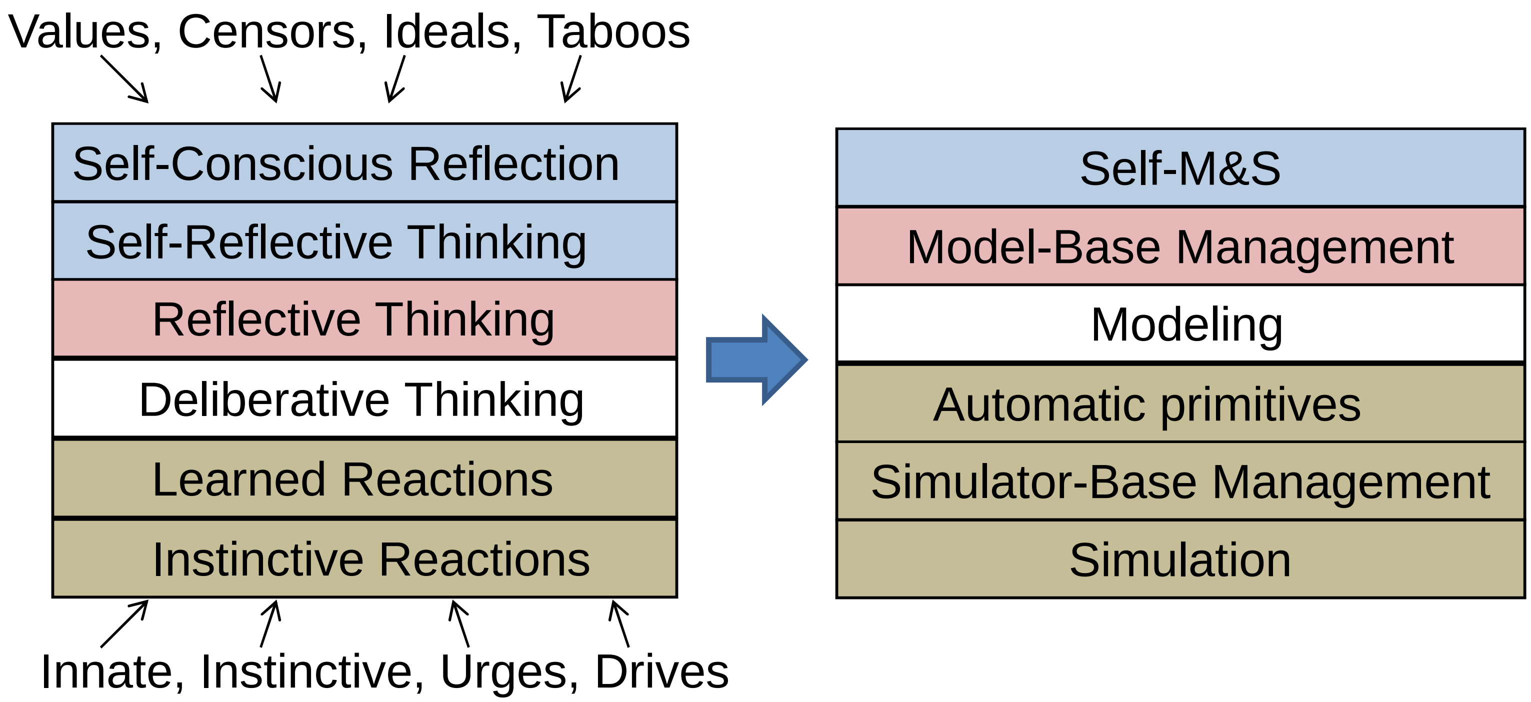
Increasing number slows down- so credit goes up as slow down – good for "Too Early" situation

Increasing number speeds up - so credit goes up as speed up – good for "Too Late" situation



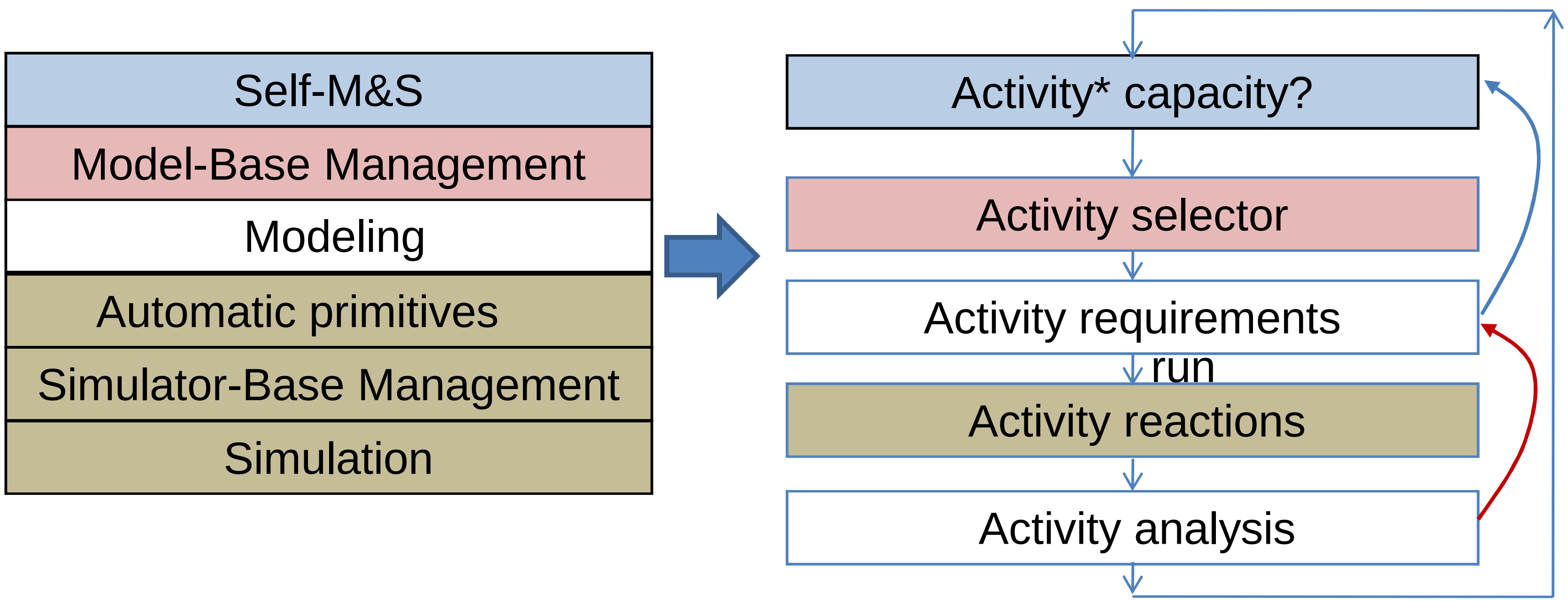
Mind Awareness
Mind Memory
Mind Decision
Intrinsic/physiological automatic mechanisms
Self-M&S
SES & Model-Base
Model & EF
Primitive/innate models & EF
Simulator-Base Management
Primitive/innate simulators
Partial coupled models
Quantized integrators
Activity-based learning
Timing properties
Synchronization
Abstract simulators
Activity tracking

Body-Brain-Mind M&S Architecture



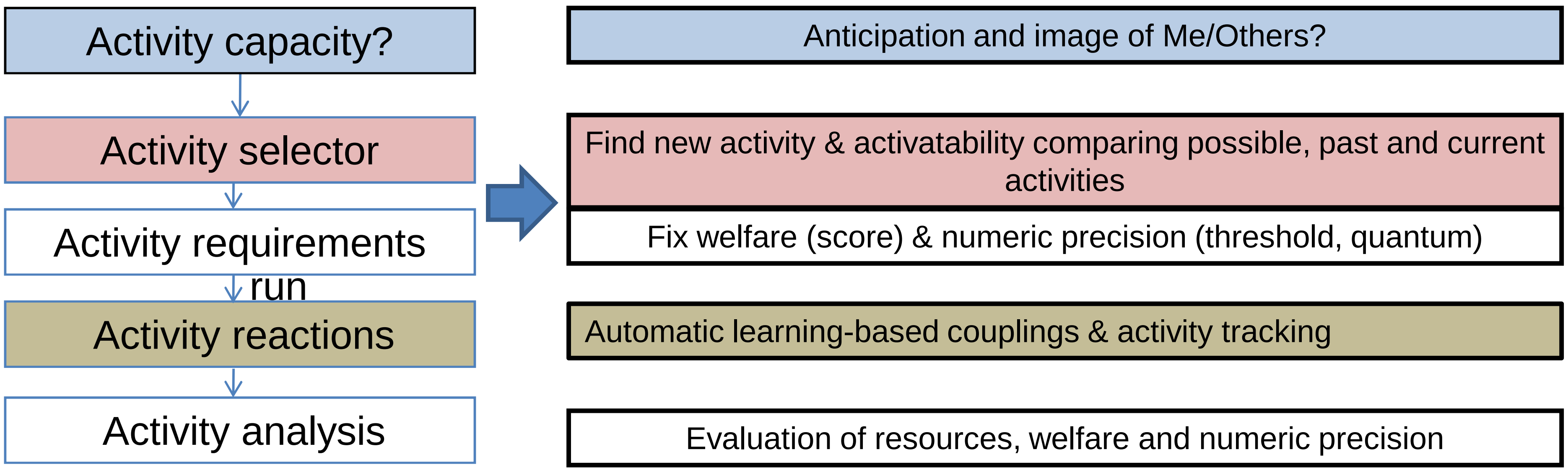
Minsky's mind architecture **Mind + Brain + Body**

Body-Brain-Mind M&S Architecture

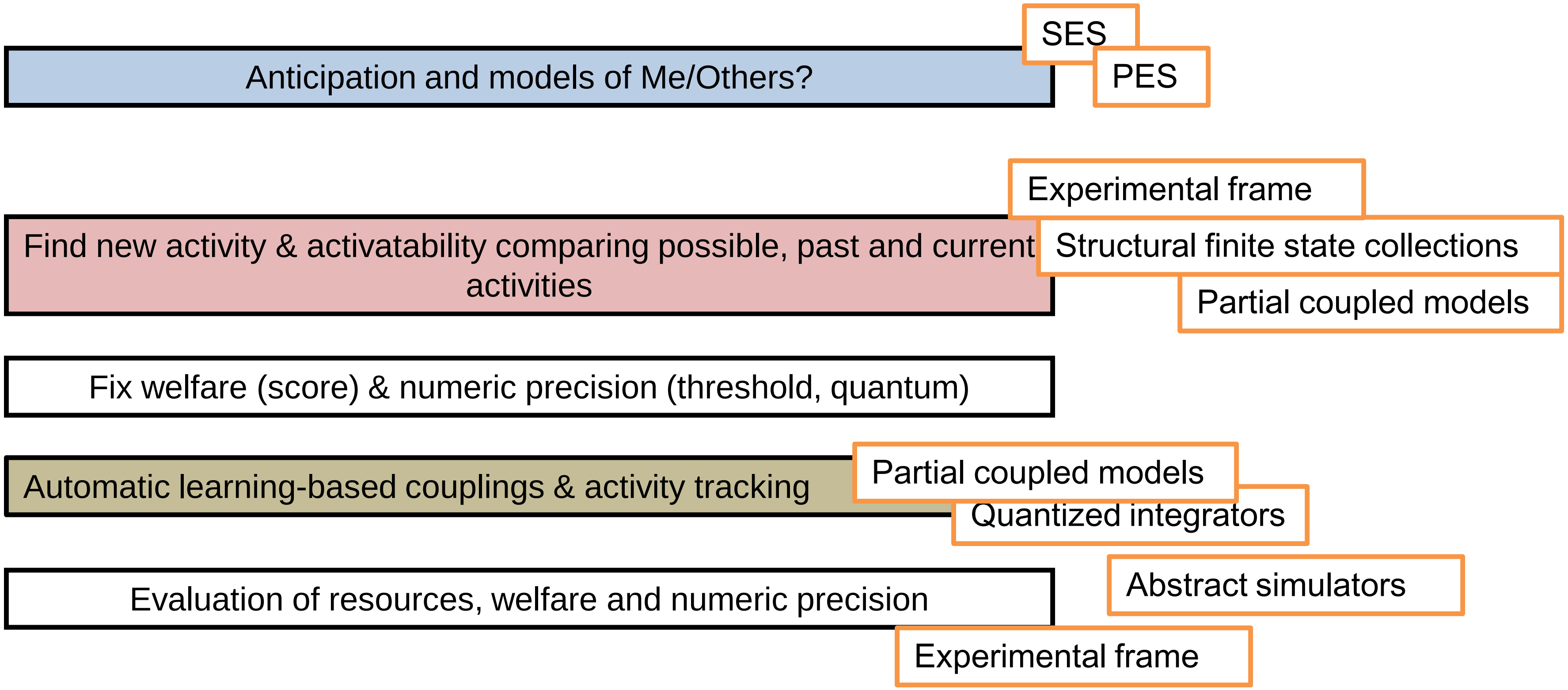


*Quality & energy

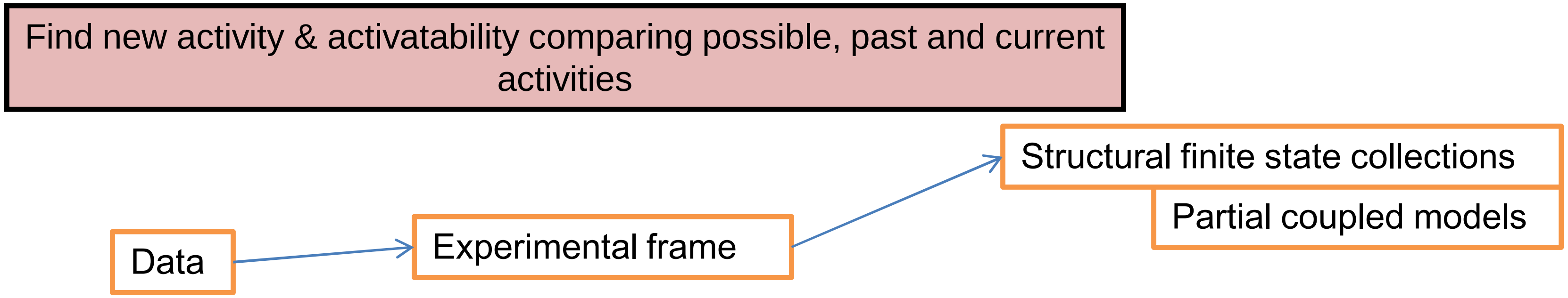
Body-Brain-Mind M&S Architecture



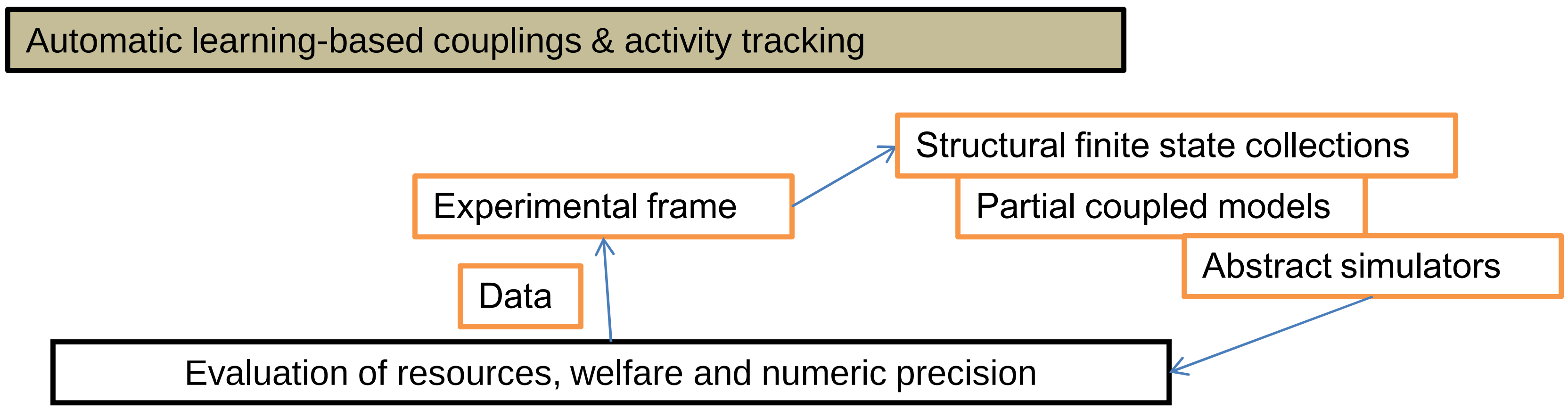
Body-Brain-Mind M&S Architecture



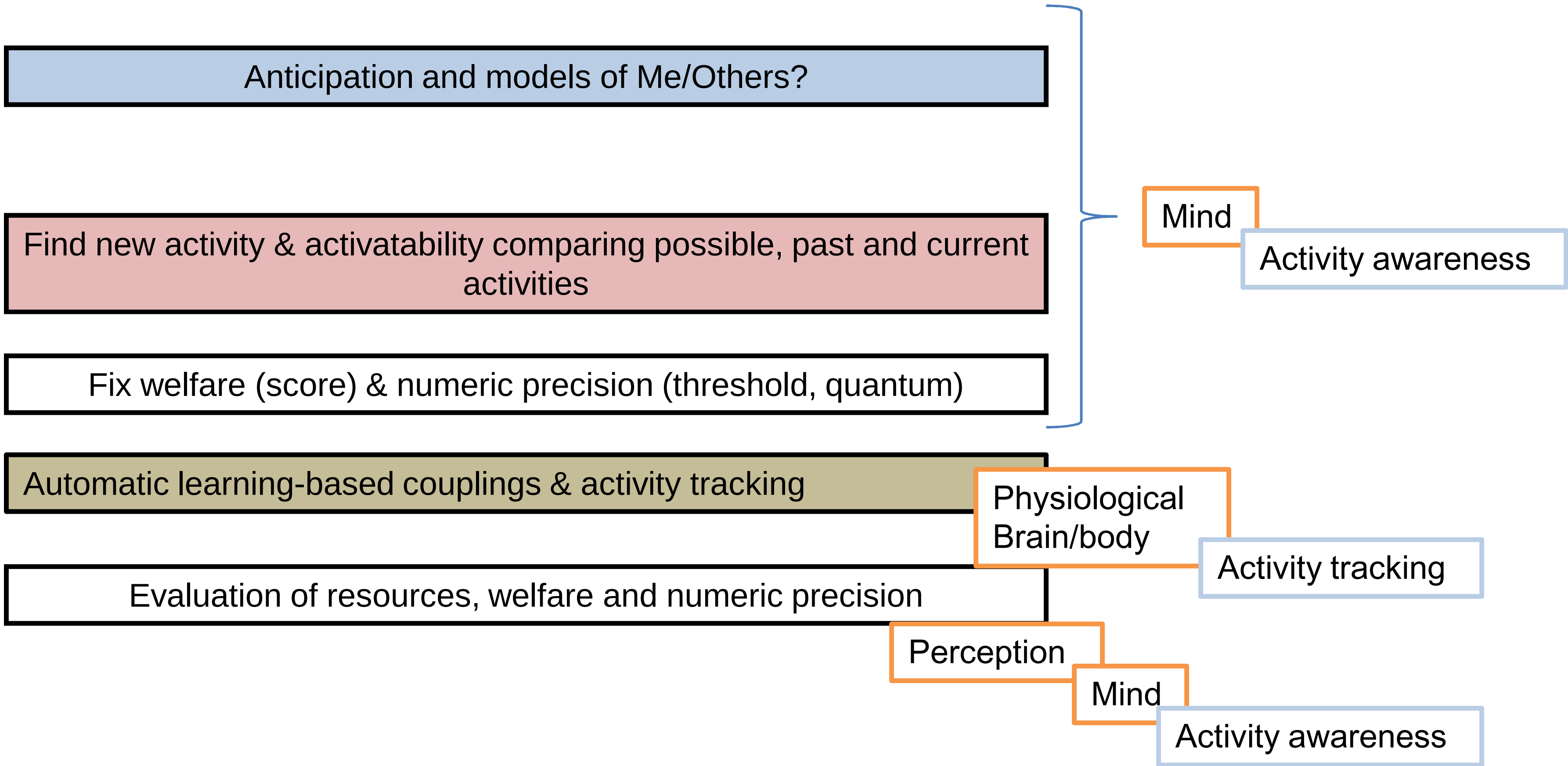
Body-Brain-Mind M&S Architecture



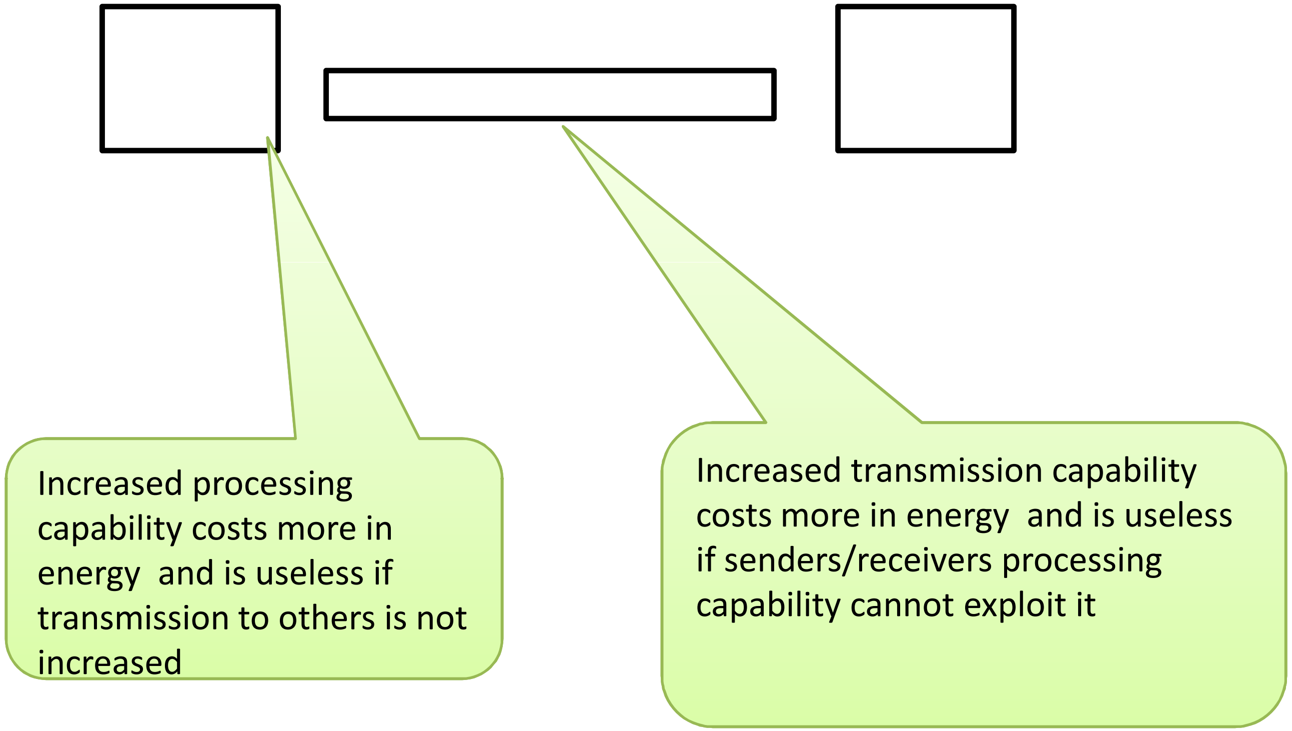
Body-Brain-Mind M&S Architecture



Body-Brain-Mind M&S Architecture

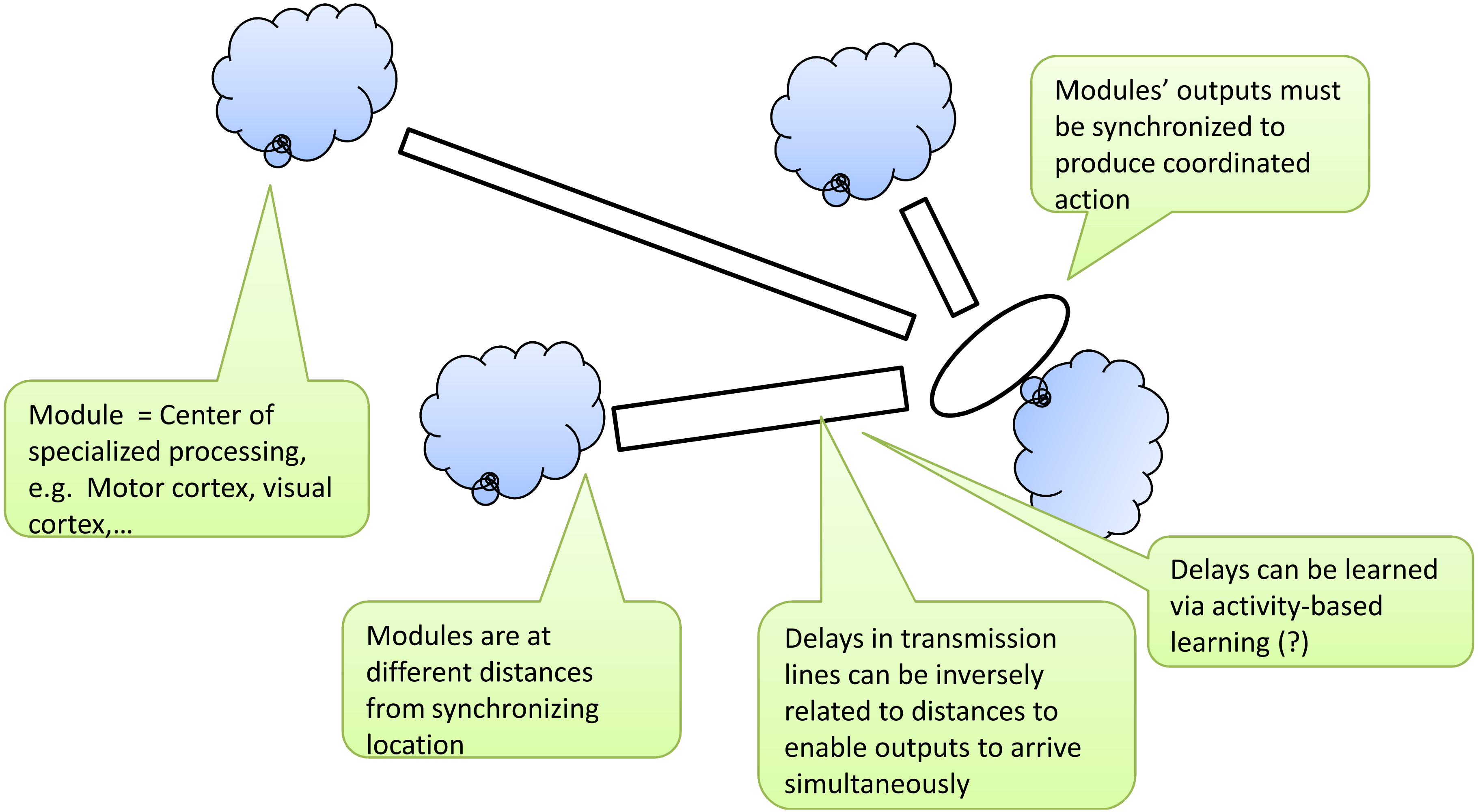


Transmission and Processing must be in balance



•Uncorrelated increases in processing and transmission will fail – unless they freeload on other adaptive improvements
•Corresponds to increased transmission capability of white matter as brain matures throughout youth
• R.D. Fields, “White Matter Matters”, Scientific American, March, 2008, pp. 54-61

Transmission delays in skill coordination



Interoperation vs Integration*

Interoperation of system components

- participants remain autonomous and independent
- loosely coupled
- interaction rules are soft coded
- local data vocabularies persist
- share information via mediation

Integration of system components

- participants are assimilated into whole, losing autonomy and independence
- tightly coupled
- interaction rules are hard coded
- global data vocabulary adopted
- share information conforming to strict standards

reusability
composability

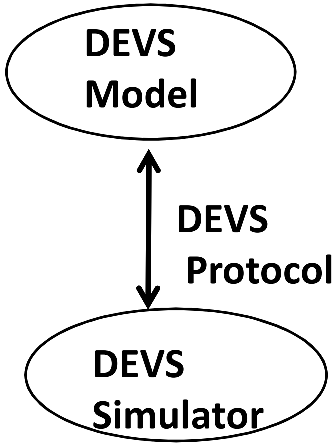
efficiency

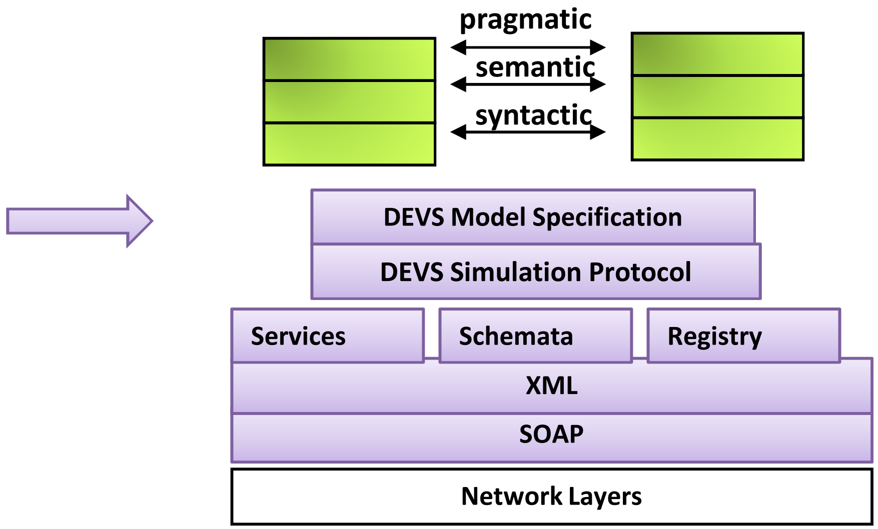
NOT Polar Opposites!

* adapted from: J.T. Pollock, R. Hodgson, “Adaptive Information”, Wiley-Interscience, 2004

DEVS Standardization Supports Higher Level Web-Centric Interoperability

DEVS Simulation Concept





DEVS Protocol specifies the abstract simulation engine that correctly simulates DEVS atomic and coupled models

- **Gives rise to a general protocol that has specific mechanisms for:**
- **declaring who takes part in the simulation**
- **declaring how federates exchange information**
- **executing an iterative cycle that**
  - ✓ **controls how time advances**
  - ✓ **determines when federates exchange messages**
  - ✓ **determines when federates do internal state updating**

Note: If the federates are DEVS compliant then the simulation is provably correct in the sense that the DEVS closure under coupling theorem guarantees a well-defined resulting structure and behavior.

•N inputs , m outputs,
• the max score is n when every input is mapped to the correct output
•there are (n*m) couplings initially,
•requiring at most 2^(nm) evaluations required for exhaustive search.

•start with the initial set of all couplings of size nm

At each stage, I,
•reduce the subset by one, i
• looking at most through each of the (ni-1) subsets

• without using component achievements vs with using component achievements

•Can show that the expected search takes time n^3 vs n^2 for
• at that stage (size ni) which adds to about (nm)^2 -- this is less then exhaustive search and made possible by the fact that only the best subset needs to be found at each stage (depends on the evaluation function). When activity-based achievements of individual couplings are used, we order the next level subsets by the total achievements and after a few stages, this results in getting the best one on the first try. So this amounts to about nm evaluations. But also for
m outputs, we simulate for about nm execution time, so the first takes about (nm)^3 versus the second (nm)^2. The hardest is when m = n and we have n^3 vs n^2. I have tried up to n = 9 and found this to be verified. But like you say, this will all depend on the particular task and algorithm used - the point is activities may be able to accelerate any such search (learning or evollution process).

On the coord and EF -- the coord works under the control of the search algorithm -- and at the end of a simulation the EF gives the result to the coord to pass on the search (actually in my current implementation it can bypass the coord -- the point is the same, the sim output needs to pass to the search algorithm).

Properties of Activity feedback for the evolution/learning

- Activity measurement – resource consumption
- Localizable in discrete units – modules
- Memorizable – activity patterns can be stored and retrieved
- Reactivatable – modules in retrieved pattern can be re-activated under control of experience – evolution, learning

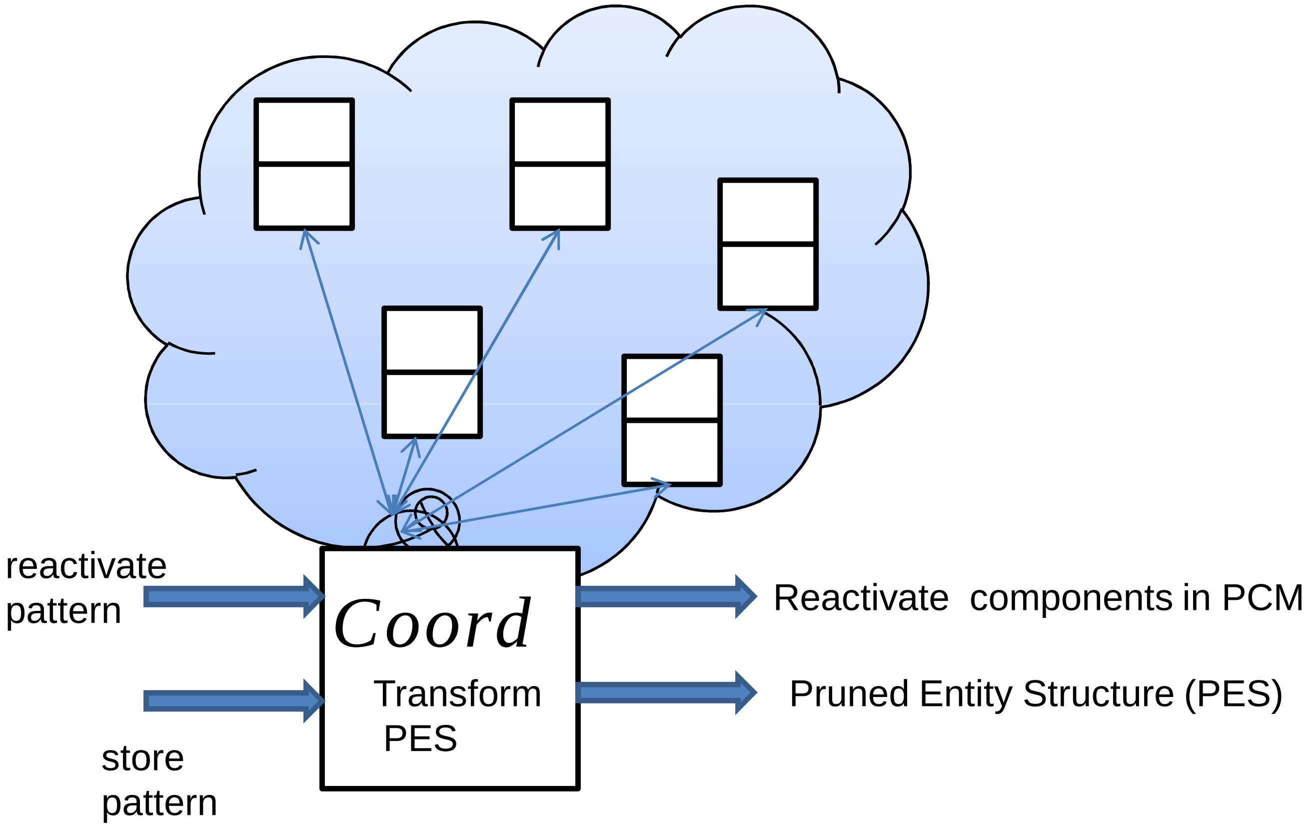
Properties interpretation

Property	Brain Evolution	Brain Learning	DEVS Formulation
Activity measurement	Energy consumption	Energy consumption	Based on simulator/ coordinator
Localizable units	neurons	neurons	Atomic and coupling components
Memorizable	Genetic memory	More activity draws more energy and increases responsiveness	Coupled models (patterns) stored in SES/PES representation
Reactivatable under control	Greater success at capturing energy enhances reproduction	Greater responsiveness increases ability to be reactivated by sensory input, activation from others and success feedback	Transformable back to executable DEVS

Candidate Coupled Models

- Let couplings be represented by components with transmission behavior
- Candidate coupled model is a set of behavior components and coupling components
- Behavior of candidate may not be efficient, may not fit behavior to be learned

Coordinator supports storage and reactivation of PCM



Store/Reactivate/Learn

- Store pattern – at the end of a trial, extract all active components (modules and couplings with activity > threshold); call this the PCM and save it in the form of a PES (XML instance) in association with the problem description
- Reactivate pattern – find pattern PESs that match problem description; select and transform one back to a PCM. Embed this PCM as a subset of components in the space of all components; initialize this subset and execute against problem.
- Since problem instances vary and the initial subset can spread activation to other components, the PCM extracted at the end of a trial can be different from that at the beginning.
- After many trials, those components with sustained high activity form the core of the solution pattern

Output Evaluation, Structure Analysis

Target I/O Function

input	output
input1	output1
input2	output2

Output produced by structure for input

Structure	input1	input2
{}	{}	{}
{c11}	{output1}	{}
{c22}	{}	{output2}
{c12}	{output2}	{}
{c21}	{}	{output1}
{c11,c12}	{output1, output2}	{}
{c11,c21}	{output1}	{output1}
{c11,c22}	{output1}	{output2}
{c22,c12}	{}	{ output2}
{c22,c21}	{}	{output1, output2}
{c12,c21}	{output2}	{output1}
{c12,c21,c22}	{output2}	{ output1, output2}
{c12,c21,c11}	{ output1, output2}	{}
{c11,c22,c21}	{ output1}	{output1, output2}
{c11,c22,c12}	{ output1, output2}	{output2}
{c[illegible]12,c22,c12}	{ output1, output2}	{ output1, output2}

evaluation of output

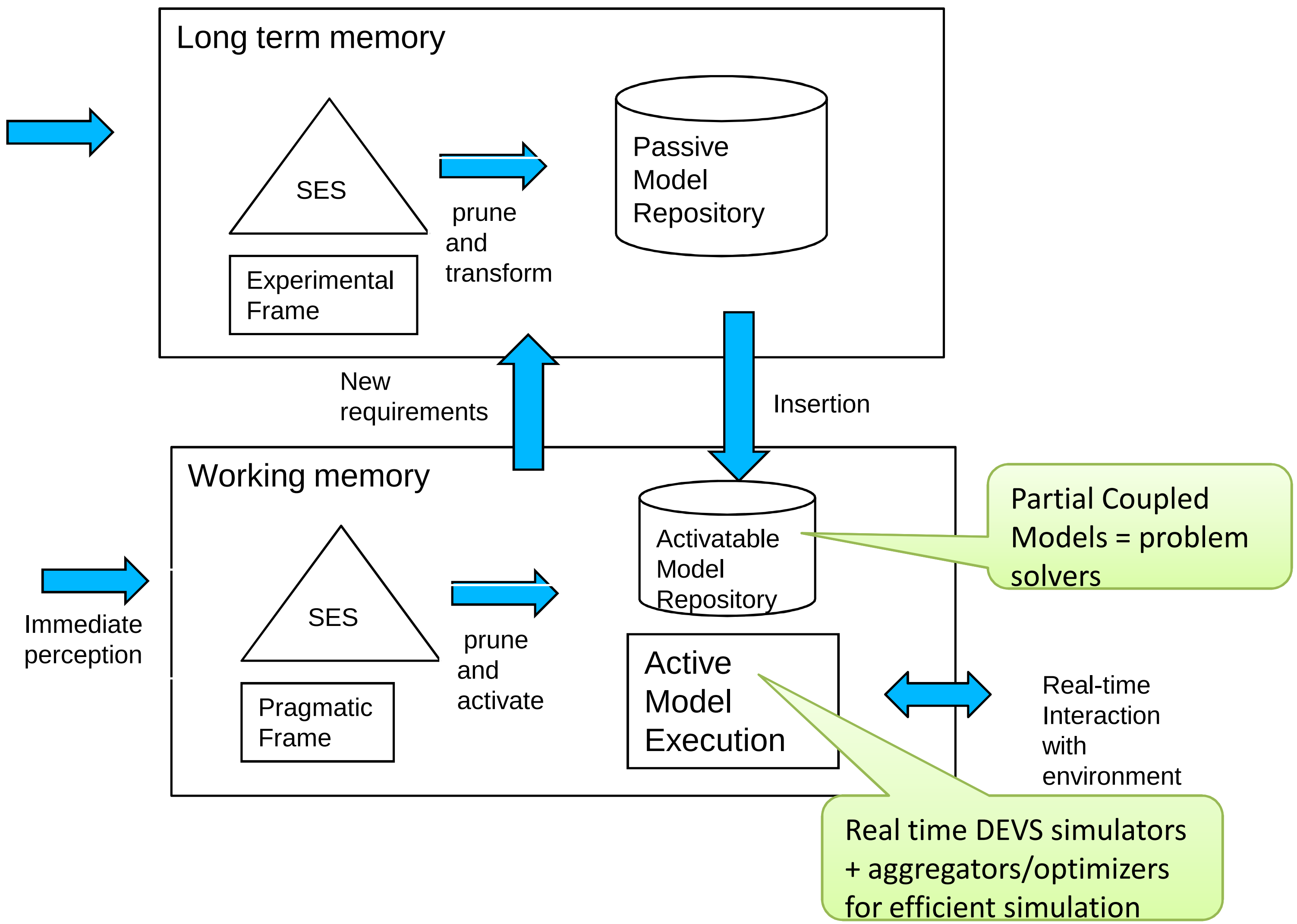
output	input1	input2
{}	0	0
{output1}	1	[-.1, 0]
{output2}	[-.1,0]	1
{output1, output2}	.5	.5

Give some credit when both outputs are produced

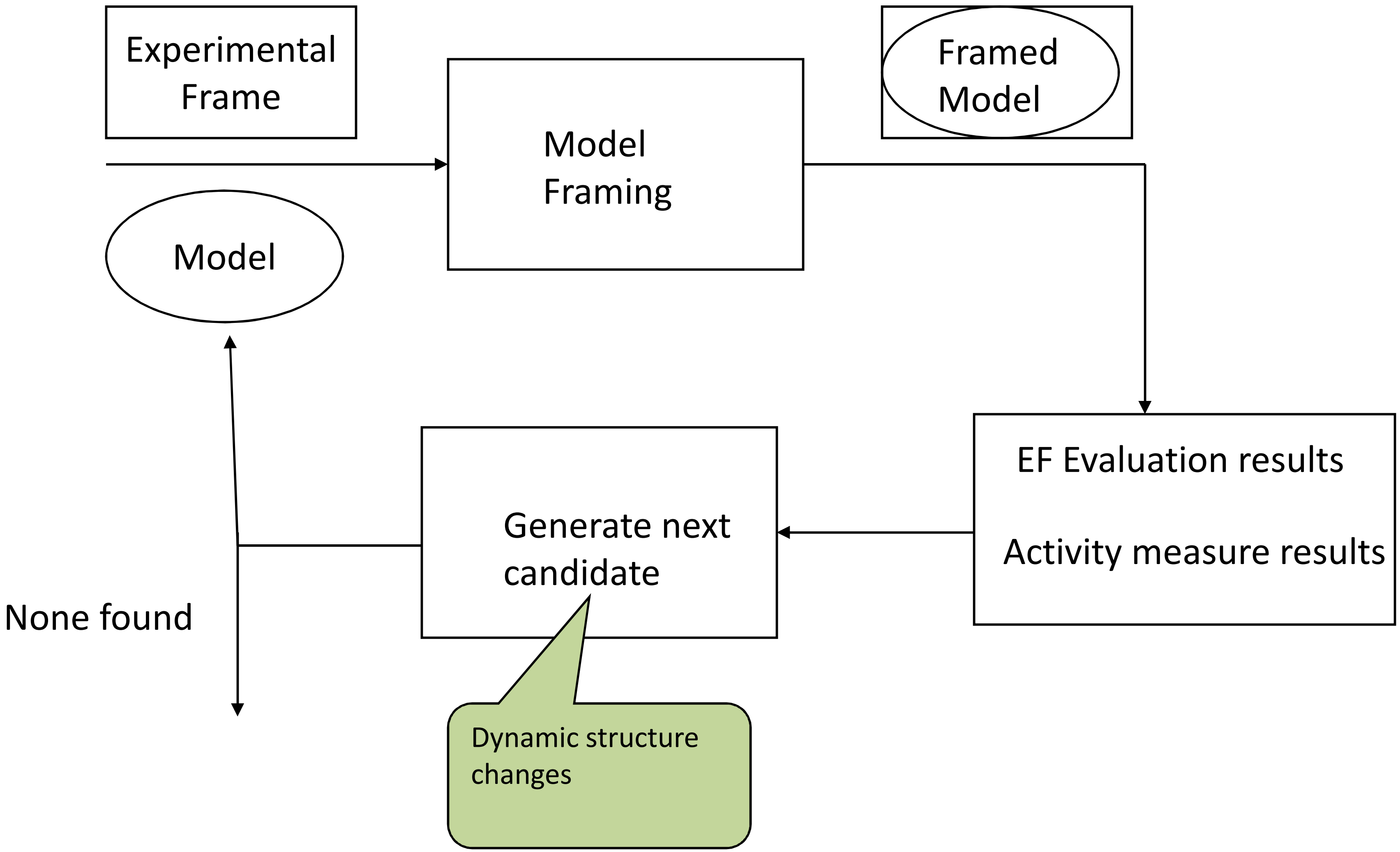
Maximum when output is correct

Give zero or negative credit for wrong output

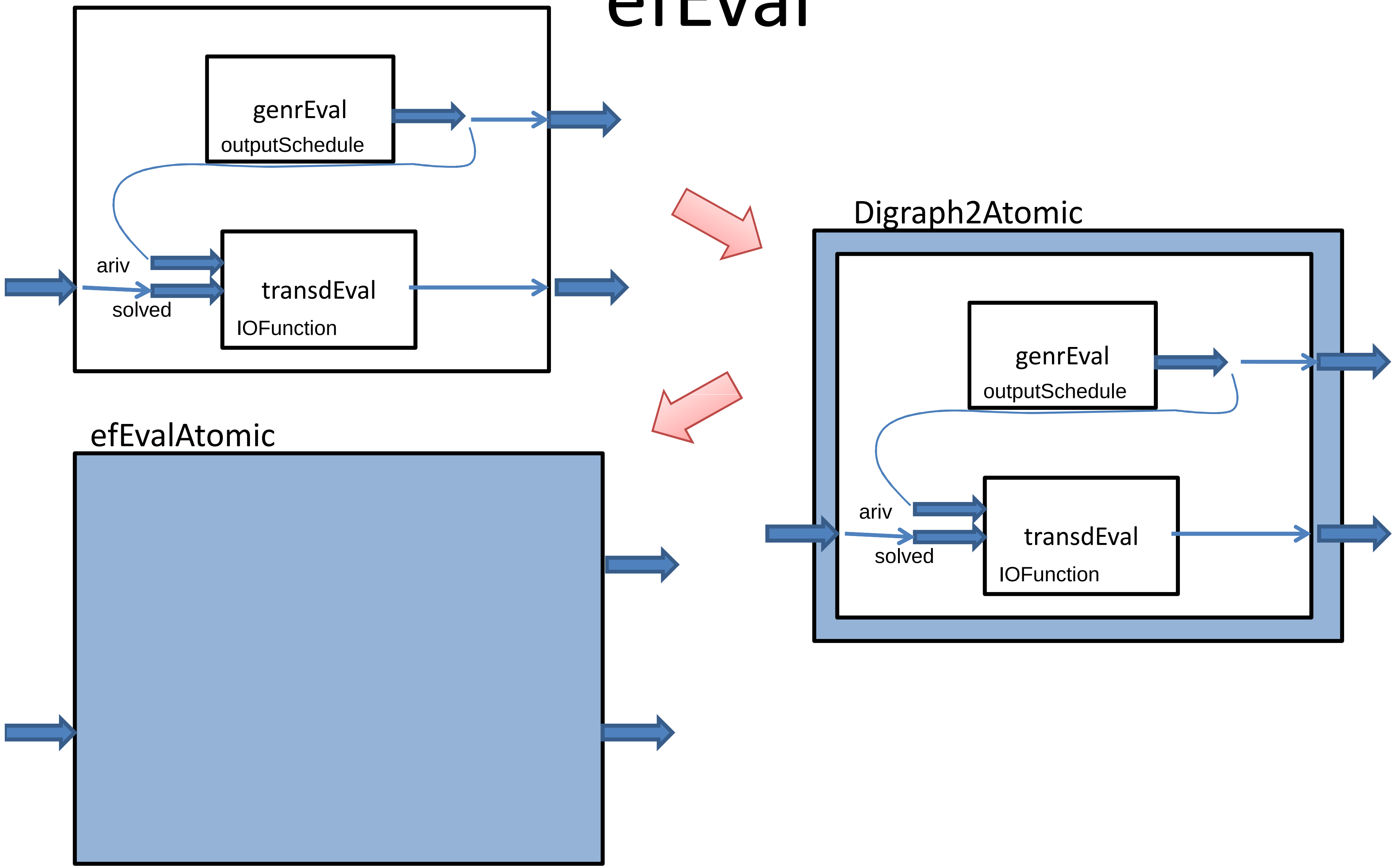
SES/Model Base Architecture for Automated M&S



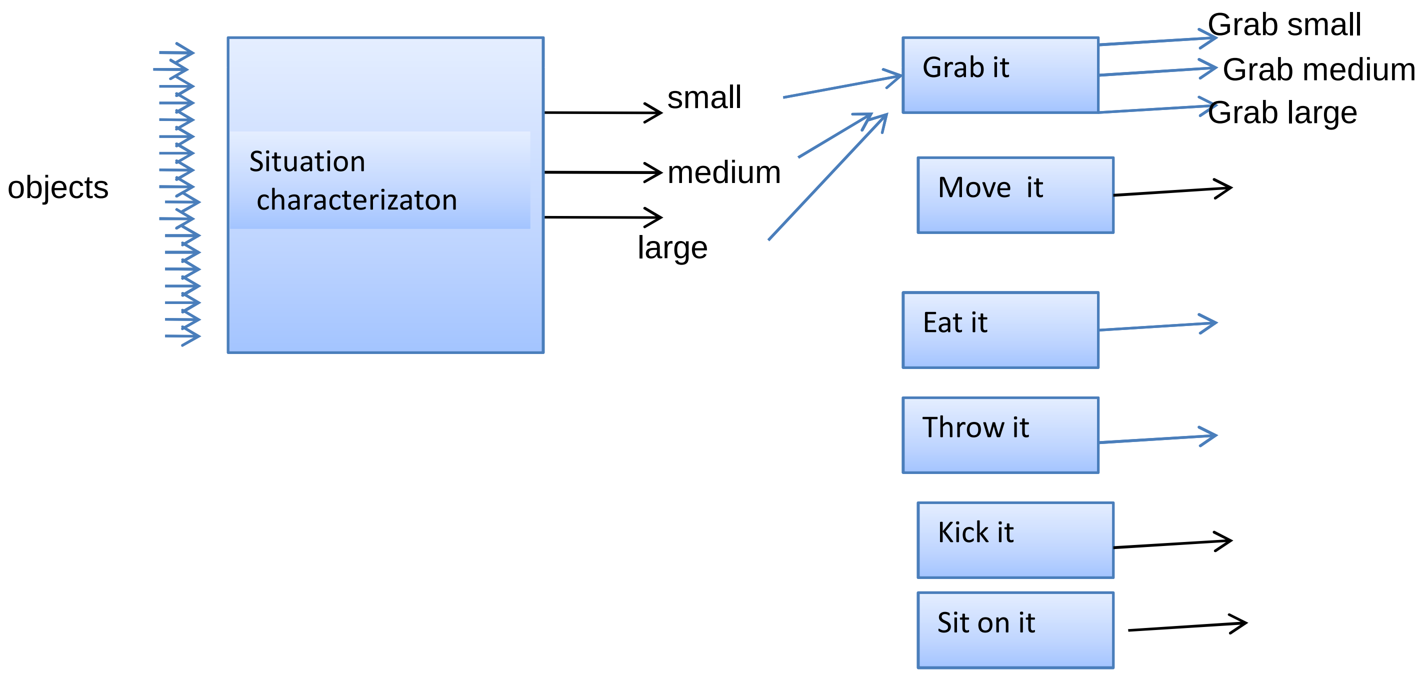
Automated Modeling Process with activity



efEval


genrEval
outputSchedule
ariv
solved
transdEval
IOFunction
Digraph2Atomic
genrEval
outputSchedule
ariv
solved
transdEval
IOFunction
efEvalAtomic

Common structure is learned whenever one of the downstream uses is activated



Common structure is learned whenever one of the downstream uses is activated