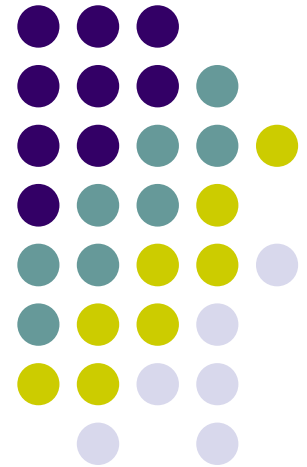
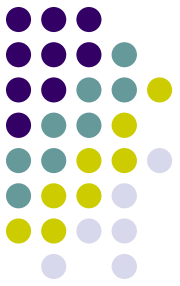


# A Two-Constraint Approach to Risky CyberSecurity Experiment Management

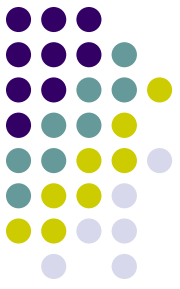
John Wroclawski, Jelena  
Mirkovic,  
Ted Faber, Stephen Schwab



# Risky CyberSecurity Research



- CyberSecurity systems becoming more complex and widely deployed
- Complexity and scale breed threats
- Experimentation as a research tool must include these features and the threats against them
  - malware
  - botnets
  - dangerous network conditions
- Goal: system to conduct risky experiments without harming infrastructure (testbeds, users, Internet)

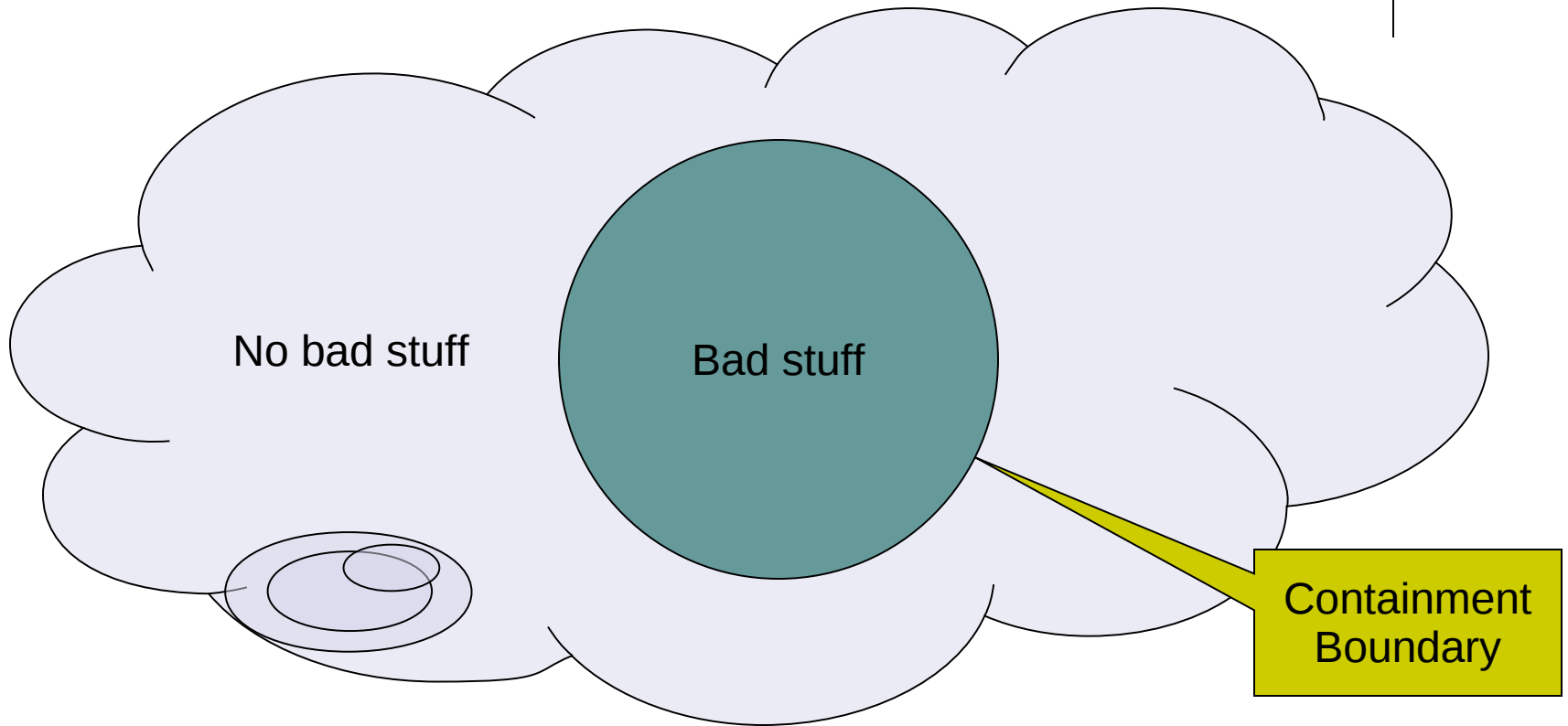
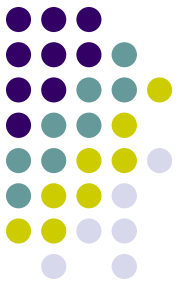


# Domains of interest

- Traditional risky experiment
  - Virus dissection
- Modern risky CyberSecurity experiments:
  - Testing malware behavior in the large
  - “Active Defense” research – embedding code
  - New defenses/new attacks – tailored malware
- All of them need to be controlled and monitored
- Shift from “Malware Containment” → “Risky Experiment Management”

# Problem formulation

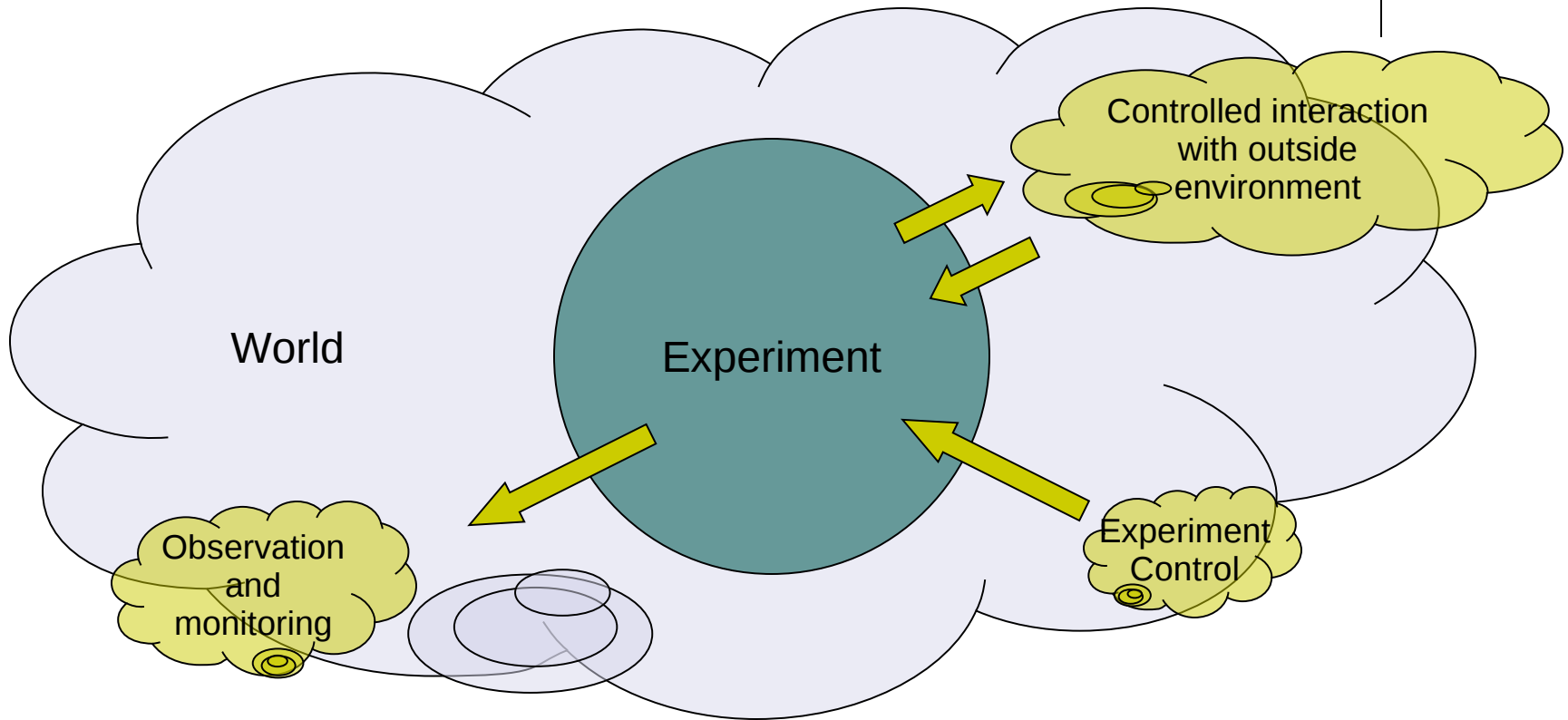
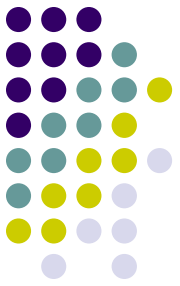
## “Classical Formulation”



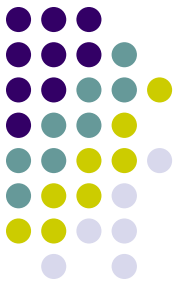
- Focus on isolation
- *De facto* emphasis of much testbed work

# Problem formulation

## More accurate formulation

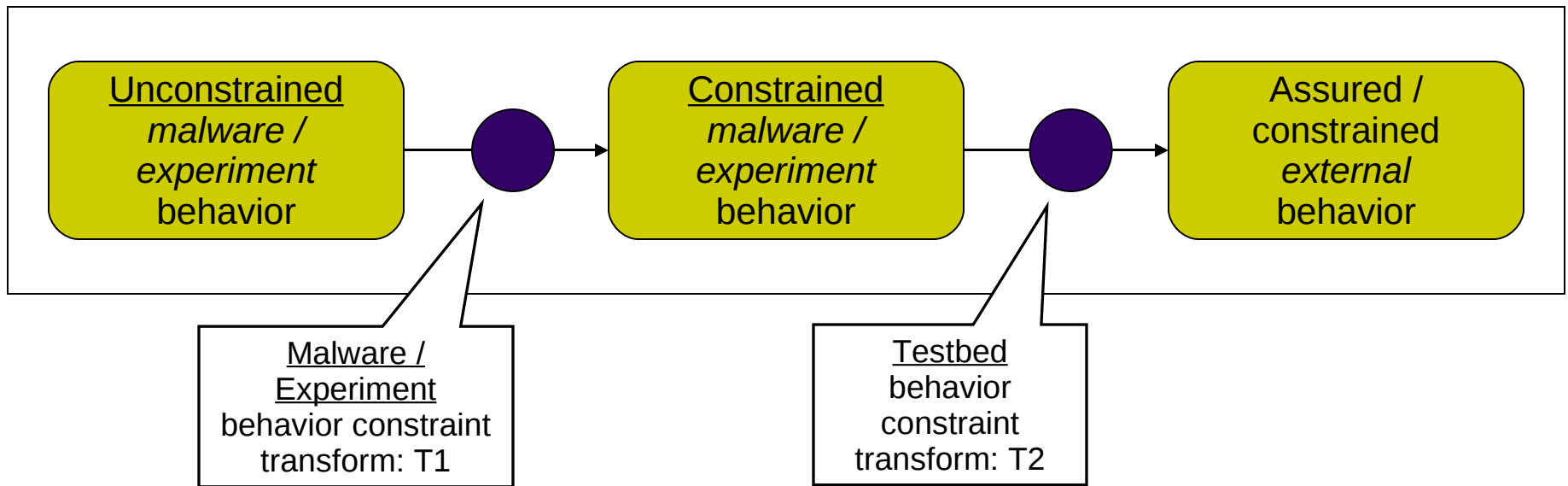


- Key issue:
  - Moving from “isolation and containment” to “understanding and assurance”

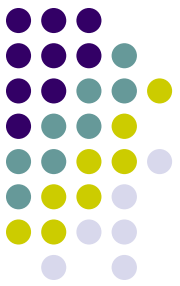


# Model

- *Two-stage* approach:



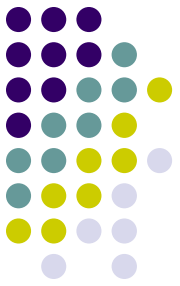
*Behavioral composition* model: External behavior == T2(T1(experiment))



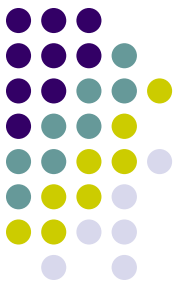
# The Power of Two Stages

- Separation of concerns:
  - Experimenter best equipped to understand impact of constraints on experiment validity - express T1 for experimenters
  - Testbed designer best equipped to understand impact of constraints on testbed and external behavior - express T2 for testbed designers/implementors
  - Each player speaks their own language
- Compositional Power
  - Constraints are synergistic, not additive
  - Composition of constraints in different domains yields expressiveness and power

# Practical Effects



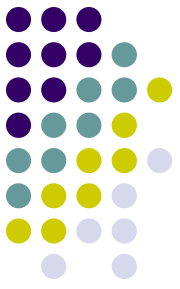
- Simplified experiment design, increased reusability
  - Defined T1 “invariants” (experiment constraints) simplify design of experiments with known external properties
- Increased assurance verifiability
  - T2 (testbed) constraints used for many experiments can be extensively tested
  - Behavior of  $T2(T1())$  may be subject to formal analysis
- Richer function
  - Simplifies reasoning about a wider range of desired behaviors
  - Simplifies tradeoff between different experimenter goals



# A (pedagogical) example

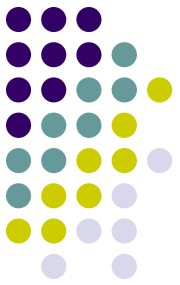
- T1: worm code constrained\* to commit suicide if it does not receive a *heartbeat* msg from specific sender every 30 seconds
- T2: heartbeat msg blocked from leaving testbed facility
- T2(T1(experiment)): worm dies if it leaves the testbed

\*we'll get to how later



## Example - part 2

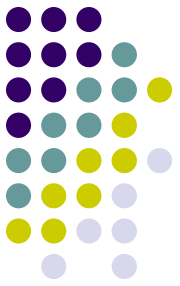
- Problem: very fast-acting worm might still spread far within 30 second constraint.
- $T1'$ : worm propagates once per heartbeat message
- $T2(T1'(exp))$ : worm dies in  $< 1$  generation if outside testbed zone
  
- Key question: does the message rate limit affect the experiment?
  - It depends.
  - That's a *good* thing!
- Feature: constraints are **explicit** and **tailorable**



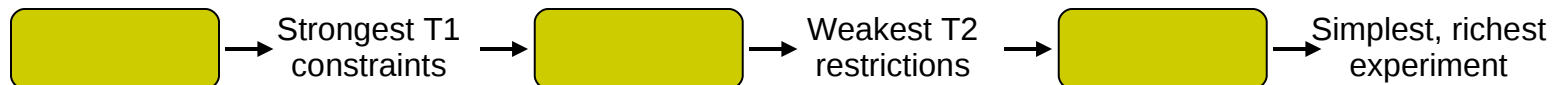
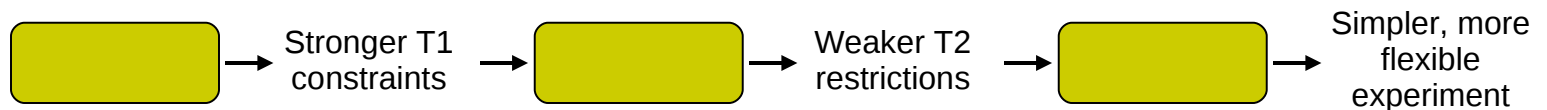
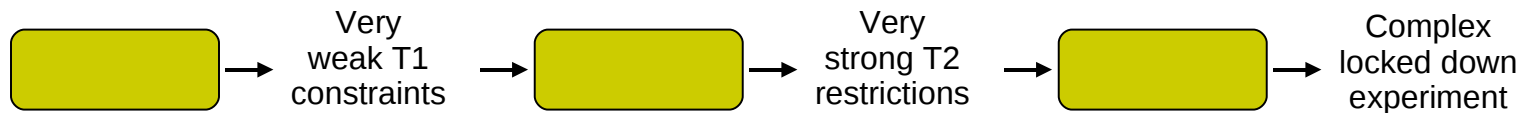
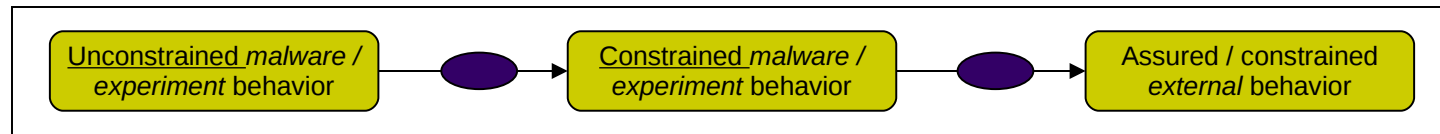
# Constraint sets

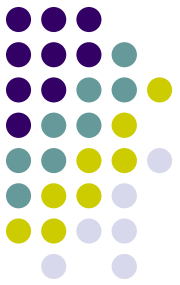
- Constraint sets are
  - Pre-established sets of T1/T2 constraints
  - Tuned for classes of experiments: Design Patterns
- To be useful, a constraint set must be
  - Useful to the experimenter
    - Some set of interesting experiments must execute correctly within the given constraints
  - Useful to the testbed designer
    - The given constraints must allow the testbed designer to ensure a desired set of external behaviors
  - Implementable
    - Have to be able to ensure that the experiment actually meets the given T1 constraints.

# Constraint Sets and Usability



- All sets meet assurance goals
- Provide a usability scale

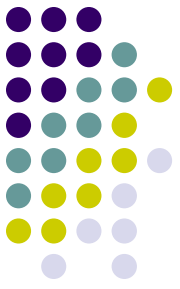




# Implementing T1 constraints

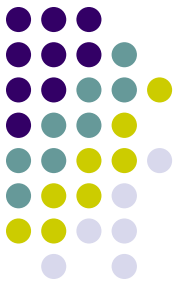
- Q1: What is the basis of the constraint?
  - Belief
  - Verification
  - Audit
  - *Enforcement*
- Q2: How might T1 constraints be enforced?
  - Wrapping
  - Code modification
  - Emulation
  - **Auditing**
- Key question: can *enforced* T1 constraints be implemented in a practical system?

# Implementation Plans: Environment



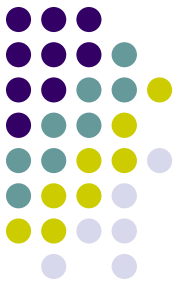
- DETER Testbed
  - USC/ISI-created/operated, Emulab-based, NSF/DHS-funded testbed
  - Collection of CyberSecurity experiment management tools
  - Community of experimenters and expertise in CyberSecurity
- More information:
  - <http://www.deterlab.net>
  - <http://www.isi.edu/deter>

# Implementation Scope



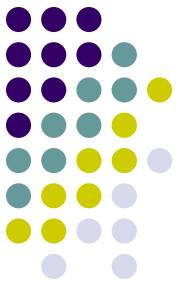
- Representative but restricted risky behavior
  - Malware
  - Disruptive behavior: e.g. DDoS
  - Access to external sites
- Categorized Risks
  - Malware disrupts later experiments
  - Experiment disrupts testbed infrastructure
  - Experiment disrupts or infects outside world

# Implementation Methodology



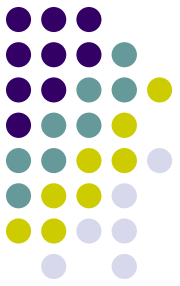
- Implement testbed constraint sets
  - Directed toward identified risk domains
  - Tight design loop with users
- Experimenters select from these implementations
- Testbed monitors and enforces their use

# Composition in Our Implementation

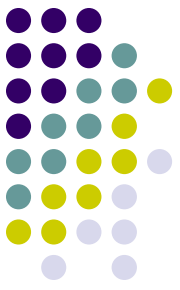


- T1 constraint: mark risky traffic
  - Malware experiment: propagation messages marked
  - DDoS experiment: attack (flood) traffic is marked
- T2 constraint options
  - Marked packets blocked
    - Malware propagation prevented
  - Marked packets rate-limited
    - DDoS Attack defanged
- Two constraint sets
  - Appropriate constraint set manages risk

# Conclusions



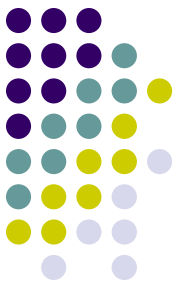
- Containment to Management
  - Enhances flexibility
  - Enables usability
- Two-level constraint model
  - Separates concerns
  - Makes constraints explicit
- Implementation to try these ideas out



# Nature of the approach

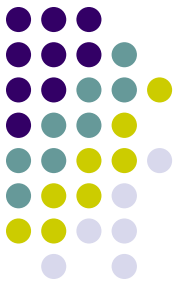
- Not an answer, but a tradeoff space
- Analogy to biological laboratories
  - Very strong containment (level 4)
    - Very complex lab
    - Very complex experimental protocol
    - Very high experimenter hassle
  - ...and vice versa (level 1)
- Proper experiment in proper lab
  - Experimenters follow appropriate protocols
  - Testbed requirements and protections tied to risk levels

# Nature of approach (analogy limits)



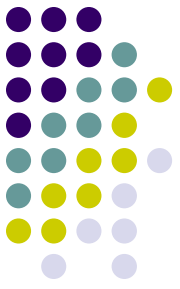
- CyberSecurity testbeds are not Bio-labs because:
  - Understanding and assurance different from containment
    - Multi-dimensional problem space
    - CyberSecurity has more attack vectors
  - Facility is more open: convenience matters
    - More dimensions, more researchers
    - Experimenters cannot specialize in testbed protocol
- CyberSecurity experimenters cooperate in **establishing** as well as **following** experimental protocol

# Beyond Implementation



- Developing REALM: language for expressing risk
- Creating REALM-based tools to:
  - Describe risk of new experiments
  - Apply T1 constraints to experiments
  - Request T2 constraints on experiments
- Integrate these tools with DETER's experiment management system: SEER

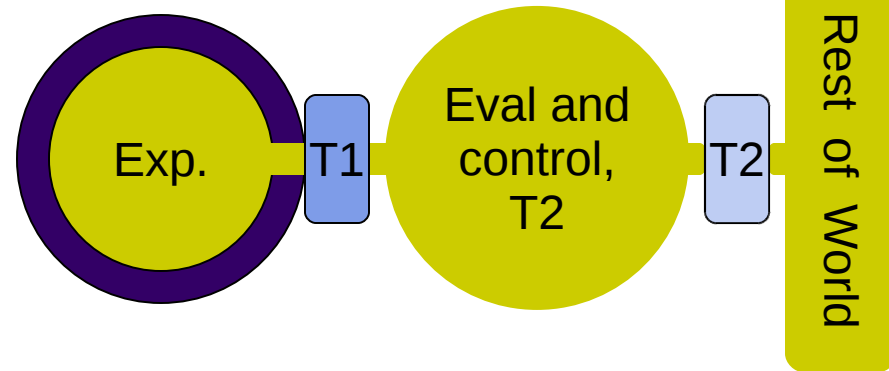
# Diversion: A challenging alternative



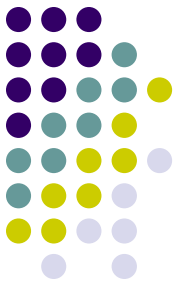
- It may be [in some cases, “is”] possible to reason formally about the overall behavior of the T2(T1exp)) system.
- This might allow *fine grain*, possibly *automatic* derivation of experiment (T1) and testbed configuration (T2) constraints to limit a particular experiment’s potential external behavior without damaging its experimental value
- Such a tool would provide a highly general facility for *limiting the risk of risky experiments*

# T1 constraint scope

- What is the *scope* over which a T1 constraint might be specified?
  - A single thread/process/host/virus/worm instance?
  - Some larger region?
- Problem *and* opportunity:
  - Composite behavior of multiple malware instances is not the same as a single instance



- Looks a bit like federation
- Looks a bit more like “classical” containment architecture
- May preserve some desirable properties of full model
  - separation of concerns
  - Modularity and defined intermediate behaviors



# Further agenda

- Motivating examples and applicability/fit with two-stage model (Monday AM)
  - Ted, Jelena, Calvin
- Overall model (Monday AM/PM)
  - Explore its value
  - Learn better how to explain it..
  - Cliff..
- Specifics and details (Monday PM)
  - Previous/early implementation approaches - Ron, Kevin
  - Constraint enforcement and validation mechanisms
  - Useful constraint sets
  - Useful external property sets
    - Protective attributes
    - Porosity attributes
- Federation, the two-stage model, and risky experiments (Tues AM)
  - Federation work progress and results
  - Possible relationship to risky experiment problem
  - Ted, Keith, Arun(?)