

Collaborators:

Norbou Buchler (ARL)

Troy D. Kelley (ARL)

Christian Lebiere (CMU)

Don Morrison (CMU)

Cleotilde Gonzalez (CMU)

Michael Yu (CMU)

Marshall Scott Poole (UIUC)

Alex Yahja (UIUC)

Tarek Abdelzaher (UIUC)

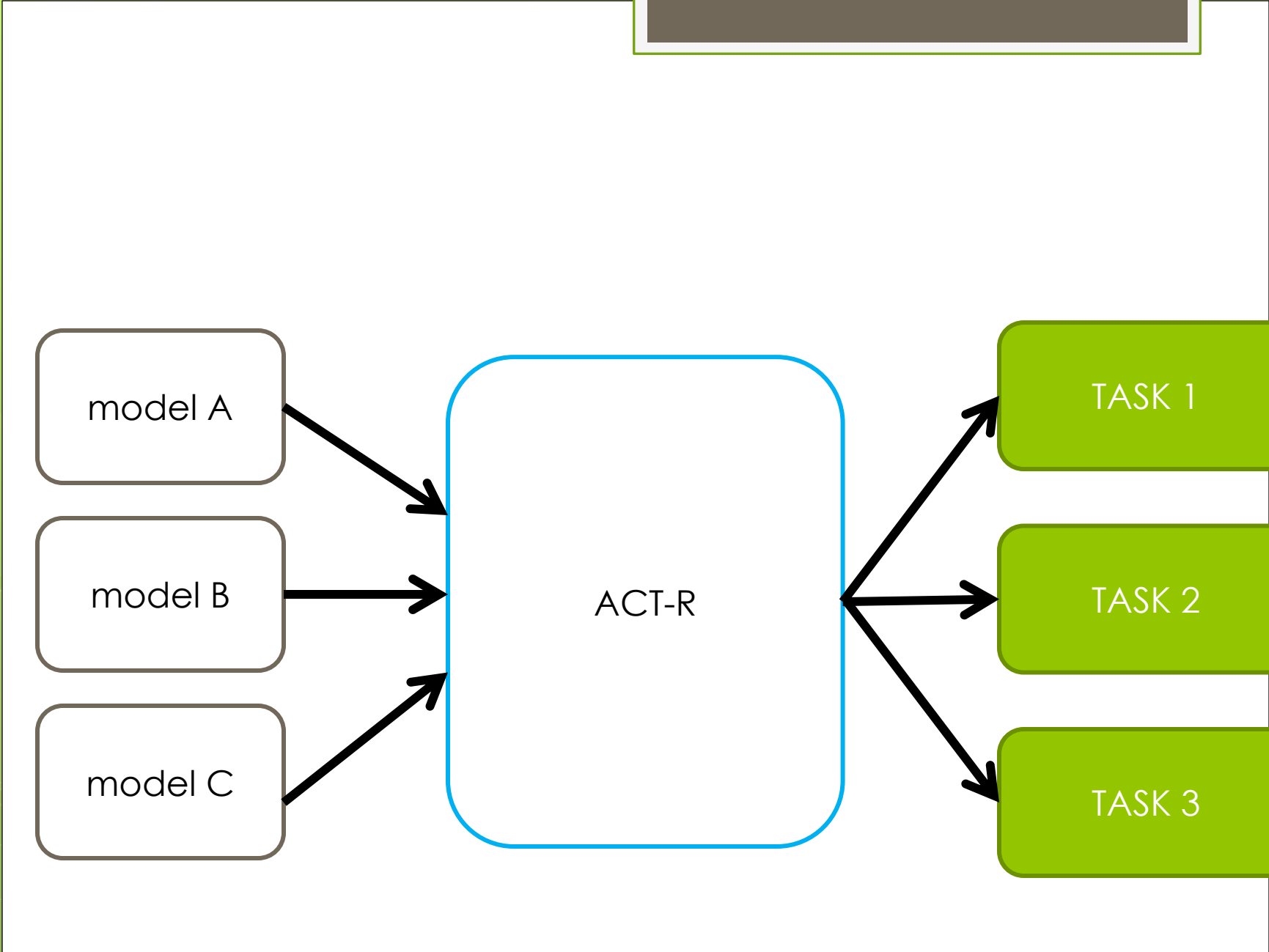
Eric Avery (ARL)

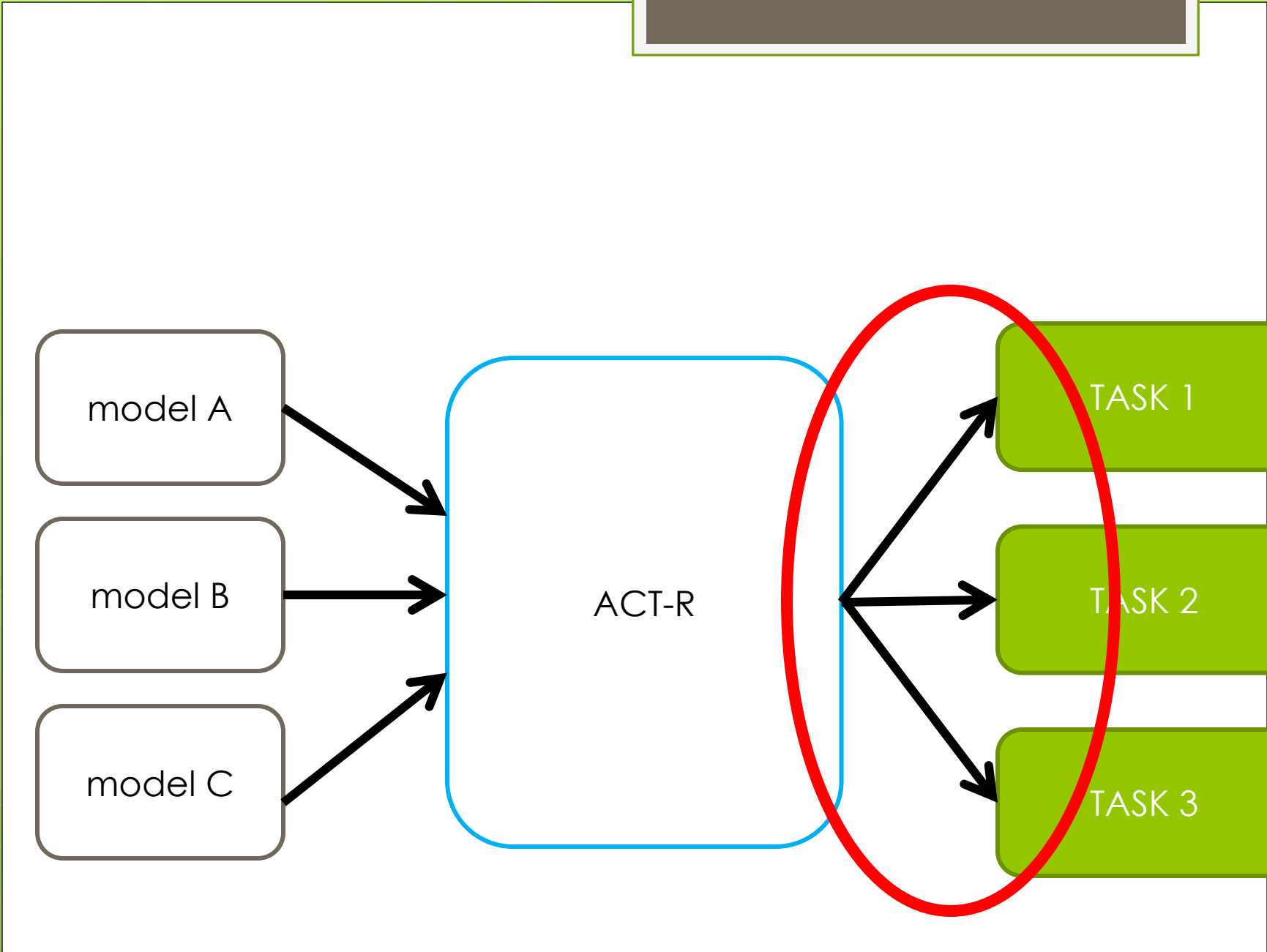
Sean McGhee (ARL)

Simple Task-Actor Protocol

Vladislav “Dan” Veksler

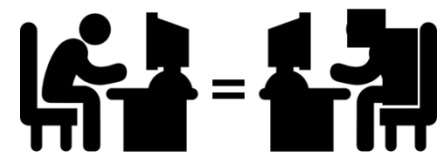






Task development

- How easy will it be to compare model data with humans?
- How easy will it be to compare model with other models?
- How easy will it be for somebody else to run their participants or models on your task?



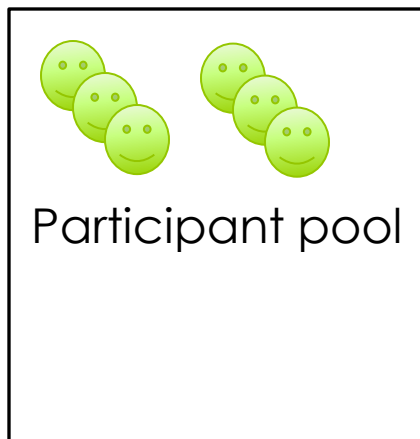
Performance Comparison

- Ideally, model performance is always contrasted with
 - Behavioral data
 - Many other computational models of varying frameworks
 - Across multiple tasks
- In reality, this is a very high hanging fruit

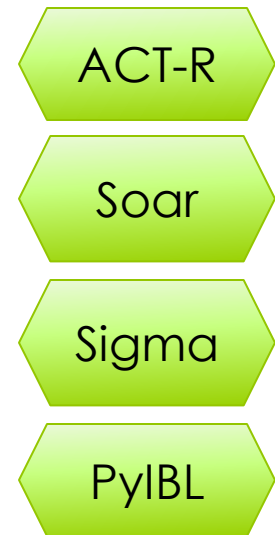
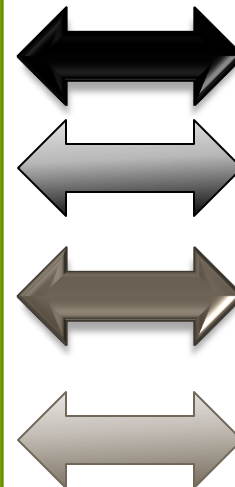
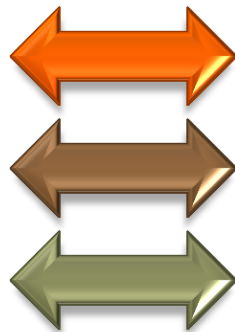
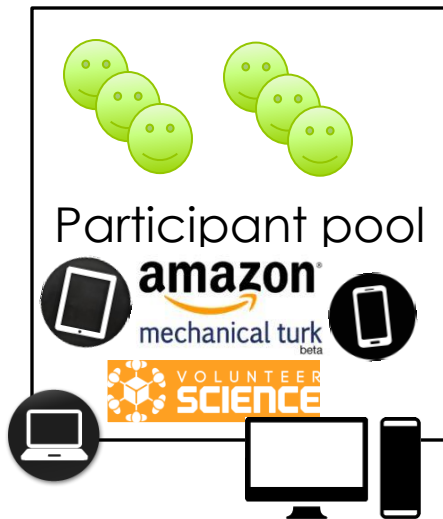
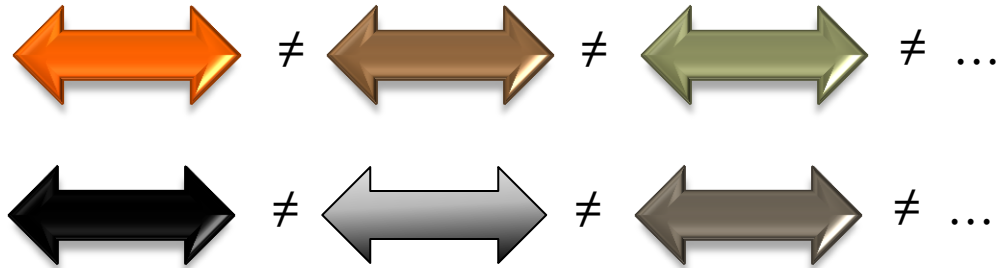
Task ↔ Actors



Loc	Att	Kind	Value	Color	ID
(25 25)	NEW	TEXT	"textBox1"	BLACK	VISUAL-LOCATION0
(35 55)	NEW	BUTTON	"button1"	BLACK	VISUAL-LOCATION1
(55 55)	NEW	BUTTON	"button2"	BLACK	VISUAL-LOCATION2



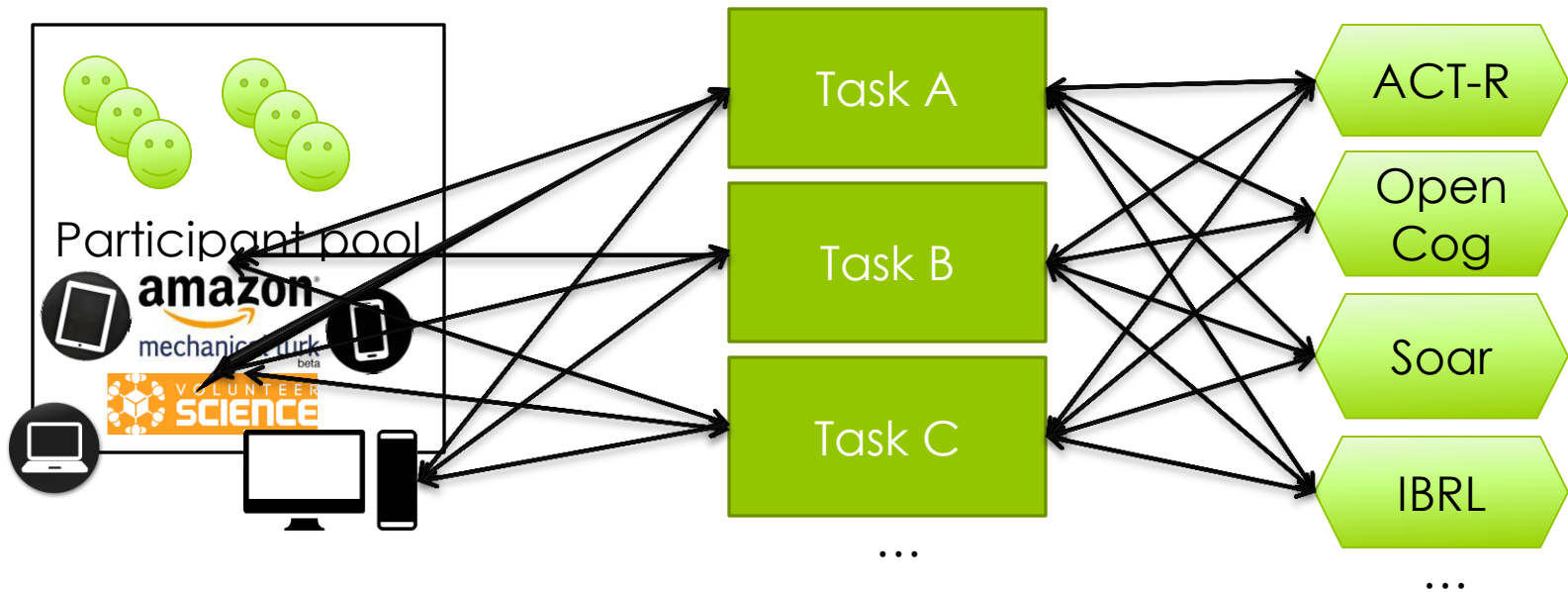
Task ↔ Actors



...

Cross-framework comparison costs become prohibitive

- Combinatorial explosion



To sum up....

- GUI \neq API
 - graphical task interfaces designed for humans are rarely machine-readable
 - task simulations designed for computational agents are rarely human-readable
- API₁ \neq API₂
 - each task simulation employs its own API
 - API designed for one agent framework is rarely readable by another

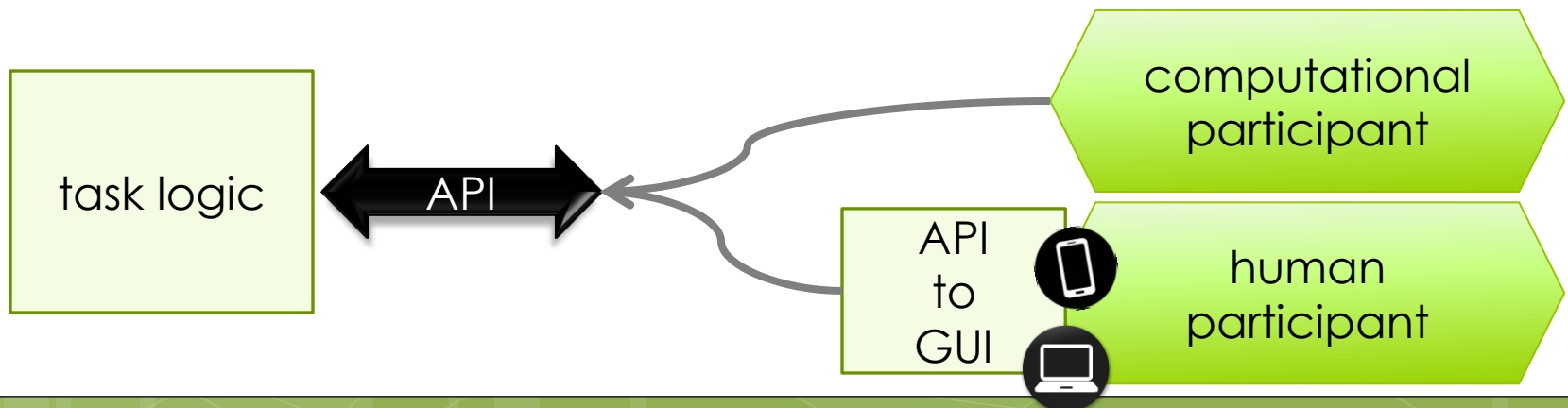
practical problem \rightarrow theoretical implications

Imagine a world where...

- you can connect your model to someone else's **plug-and-play** task software
- you can grab someone else's simulation and **replay** it in your lab without any non-standard software
- you can **replicate** someone else's experiment results without hassle
- you can **connect** different cognitive systems to your task without hassle

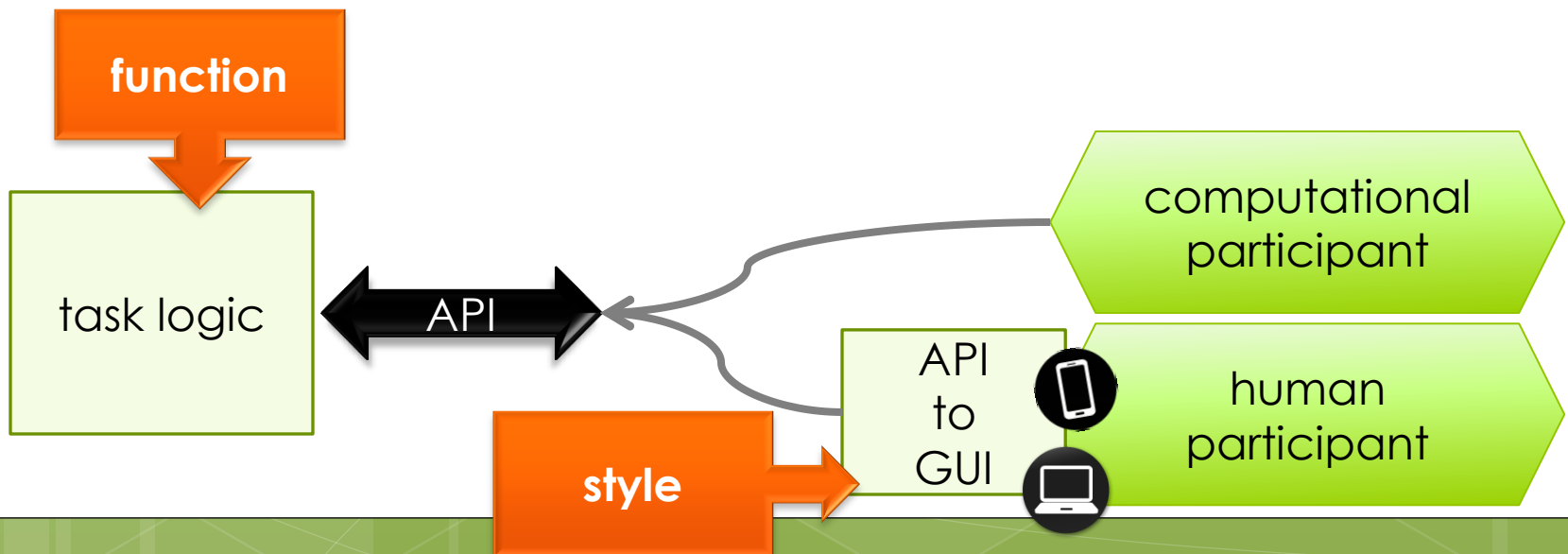
The solution

1. A functional-essence approach to task development



The solution

1. A functional-essence approach to task development
 - separation of function from style



Function VS Style

- Function
 - two numbers
 - textbox
 - submit button
- Style
 - spacing and colors
 - font-type and size
 - button style

```
<html>
<head><style>
input[type="submit"] {background-
    color:#aaccff;border-radius:5px;}
</style></head>
<body>
<table width=100% height=100%><tr
    valign=middle><td align=center>
<form style="font-family:tahoma;font-
    size:18px;">
5 + 5 = <input>
<input type=submit>
</form>
</td></tr></table>
</body></html>
```

5 + 5 =

Submit

Functional-essence approach to task development

- Function

- 8x8 table with alternating squares
- 6 different recognizable piece types



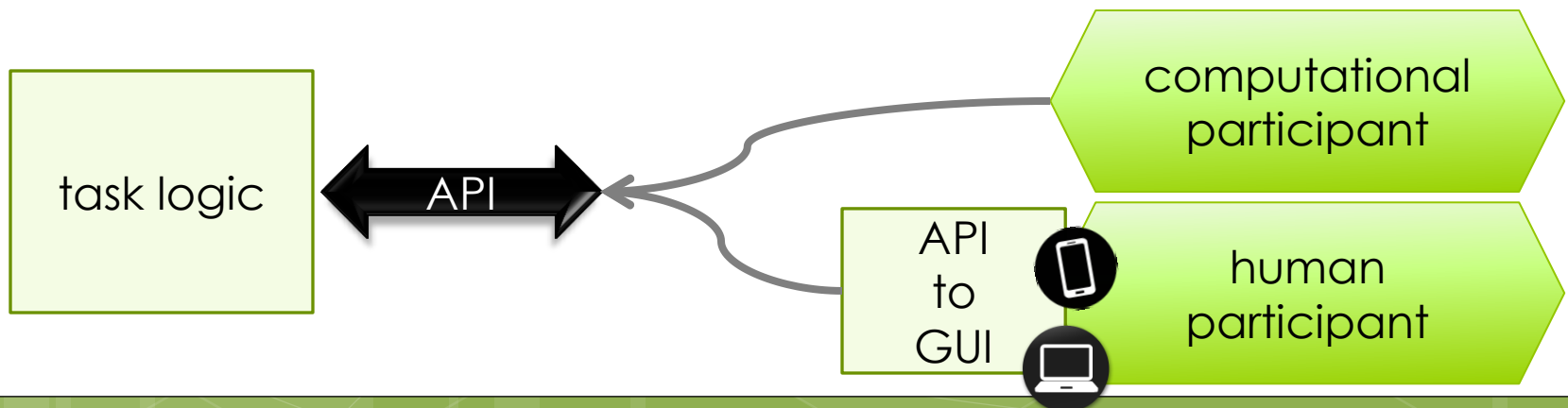
- Style

- size and colors of the board and squares
- images representing each piece



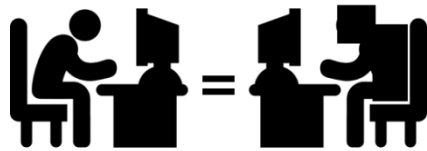
The solution

1. A functional-essence approach to task development
2. A standard & simple API for function-only interaction with most task types



Simple Task-Actor Protocol (STAP)

<http://vdv7.github.io/stap/>



Veksler, V. D., Buchler, N., Lebiere, C., Morrison, D., & Kelley, T. D. (in press). The performance comparison problem: Universal task access for cross-framework evaluation, Turing tests, grand challenges, and cognitive decathlons. *Biologically Inspired Cognitive Architectures*.

Collaborators:

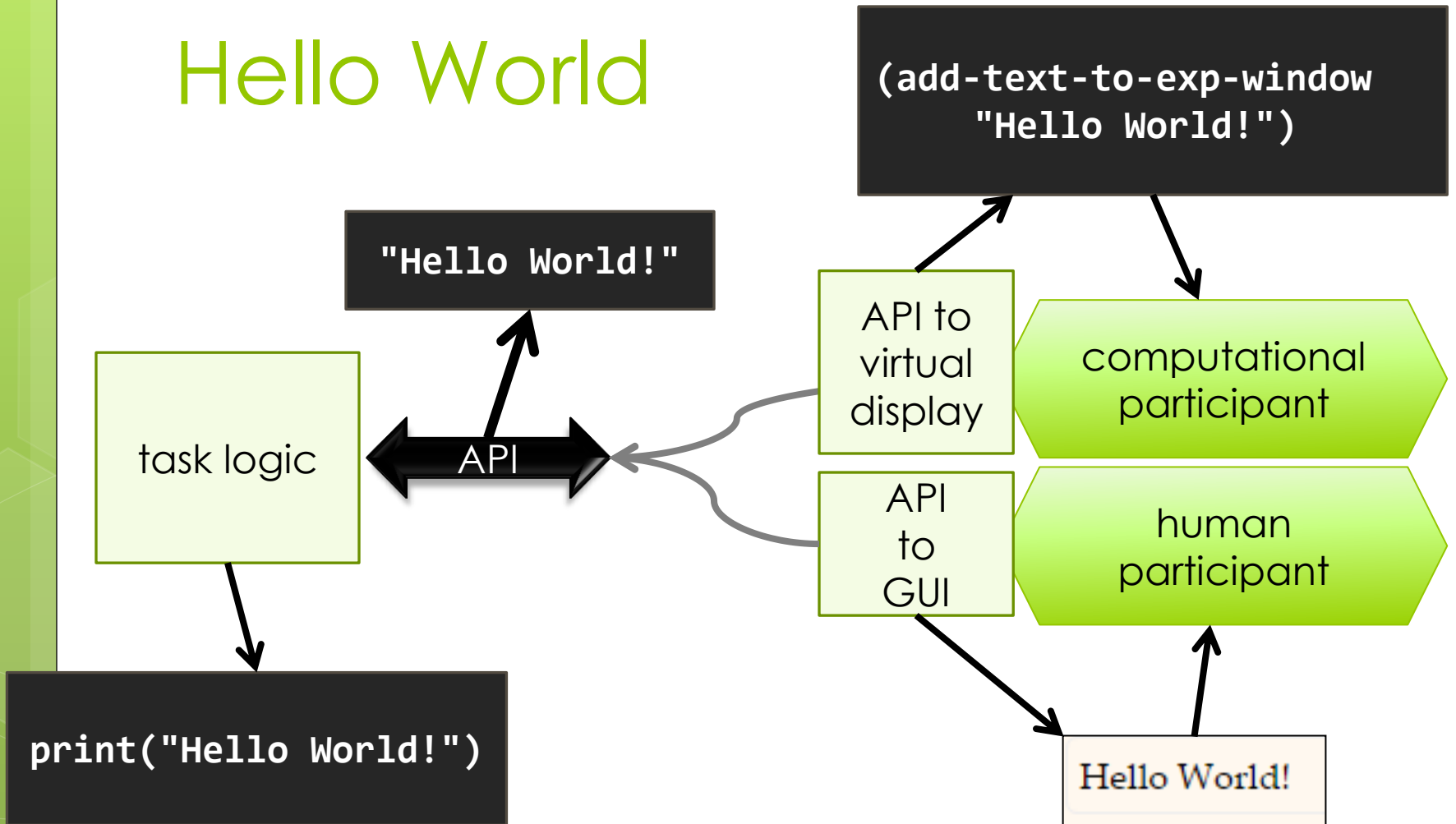
Norbou Buchler (ARL)
Troy D. Kelley (ARL)
Christian Lebiere (CMU)
Don Morrison (CMU)
Cleotilde Gonzalez (CMU)
Michael Yu (CMU)
Marshall Scott Poole (UIUC)
Alex Yahja (UIUC)
Tarek Abdelzaher (UIUC)
Eric Avery (ARL)
Sean McGhee (ARL)

Simple Task-Actor Protocol (STAP)

- **Subset of the JSON protocol**
 - JSON modules in most programming languages
- **Task interactions**
 - Vector graphics and animation
 - Varying types of actions (e.g., click, hold-down, type)
- **Time**
 - Faster-than-real-time simulations
 - Slower-than-real-time simulations
- **Goal** and task description
 - Machine-readable
 - Auto-generated human instructions

STAP

Hello World



STAP Style Templates

Hello World!

Click a button

Button 1

Button 2

Hello World!

Click a button:

[Button 1](#) [Button 2](#)

Hello World!

Click a button:

[Button 1] [Button 2]

task logic

STAP

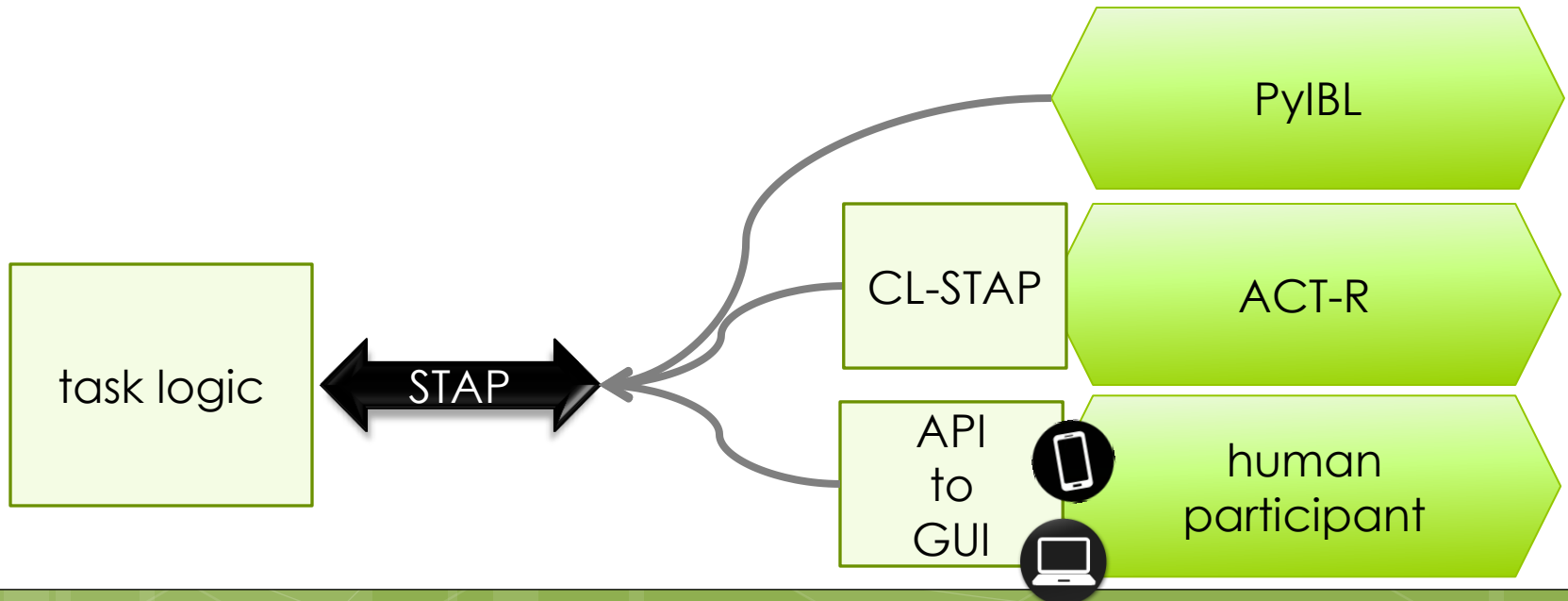
API
to
GUI



human
participant

STAP for ACT-R

- Don Morrison's (CMU)



STAP interaction/playback

- RedForce task videos
 - [ACT-R model](#)
 - [PyIBL agent](#)
 - [human participant](#)

The image displays three sequential screenshots of a STAP playback interface, showing the progression of a mission. Each screenshot includes a status bar at the top, a 'Result' section, a 'Map' section, and a 'convoy' control panel at the bottom.

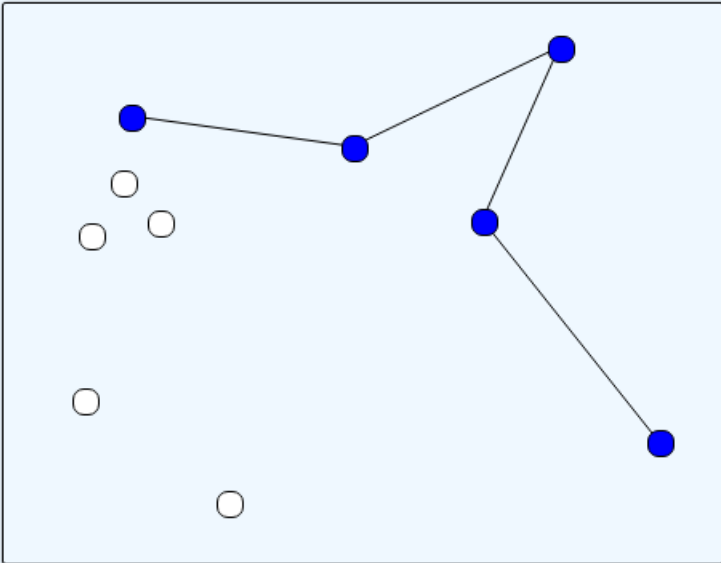
- First Screenshot:**
 - Status: Team: blue Day: 5 | blue force damage: 1 of 5 | village A sentiment: 2 of 6 | village B sentiment: 1 of 6
 - Result: convoy: delivered aid; village B sentiment: increased
 - Map: Shows a map with routes (Alpha, Beta, Gamma, Delta) and locations (Mountain Pass, Village A, Village B, Army Post, Mountain Camp, Valley Shipping Route).
 - convoy: deliver aid (highlighted in red), stay on post
- Second Screenshot:**
 - Status: Team: blue Day: 5 | blue force damage: 1 of 5 | village A sentiment: 3 of 6 | village B sentiment: 1 of 6
 - Result: digger: (digger icon); village B sentiment: increased due to safe/spy action delays
 - Intel: bombmaker location: A
 - Map: Shows the same map as the first screenshot.
 - convoy: deliver aid (highlighted in red), stay on post
- Third Screenshot:**
 - Status: Team: blue Day: 5 | blue force damage: 0 of 5 | village A sentiment: 6 of 6 | village B sentiment: 2 of 6
 - Result: convoy: delivered aid; village B sentiment: increased due to safe/spy action delays
 - Map: Shows the same map as the first screenshot.
 - convoy: deliver aid (highlighted in red), stay on post

STAP task examples

○ <http://vdv7.github.io/stap/>

- Instructions
- Produce the shortest possible circuit that connects each circle on the map.
 - Click a white circle to add it to the circuit.
 - Click a blue circle to remove it from the circuit.

Map



Trial 2 of 32

Is this object a Greeble or a Groble?



Groble

Greeble

Instructions

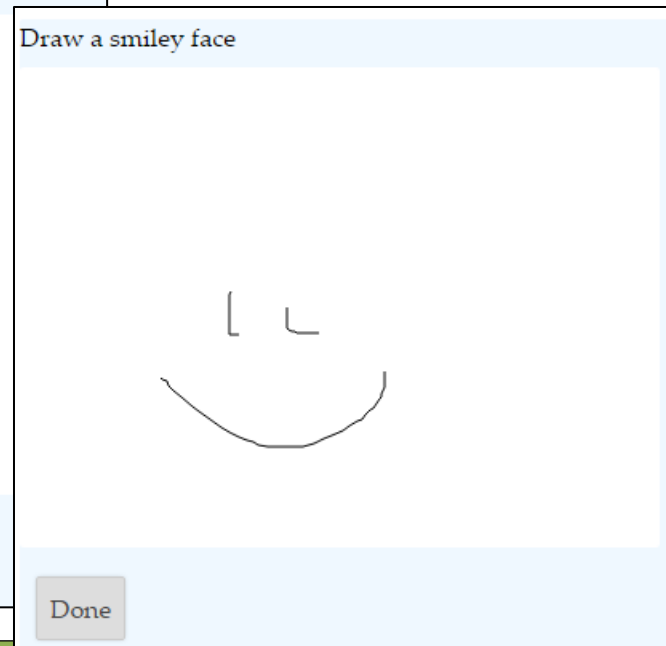
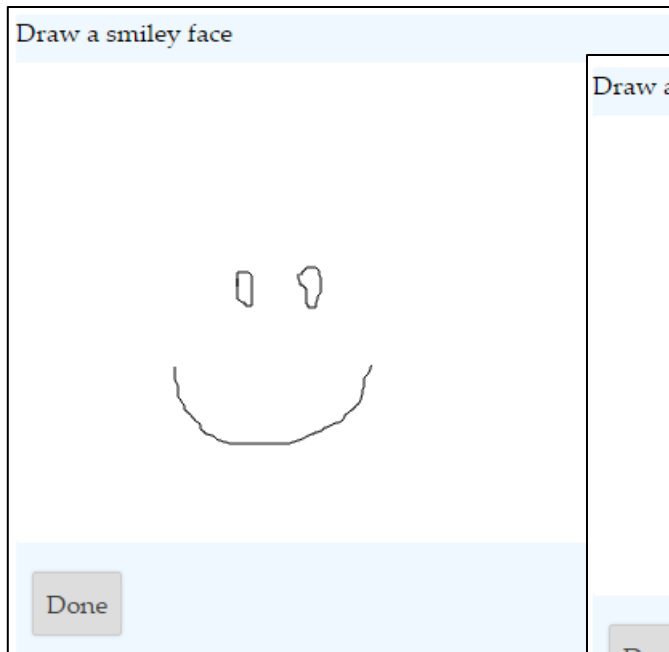
Press the "Start Timer" button to start a timer. Press "Stop Timer" when you think the ball reaches the RED line.



Stop Timer

STAP task examples

- <http://vdv7.github.io/stap/>



STAP task examples

Robot connected: Laser scanner on board. Battery: 13 of 13

Task: ss-rics

Spatial Segments



Robot connected: Laser scanner on board. Battery: 13 of 13

Task: ss-rics

Spatial Segments

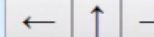


**Robot navigation:
human vs instance-based learning agent**

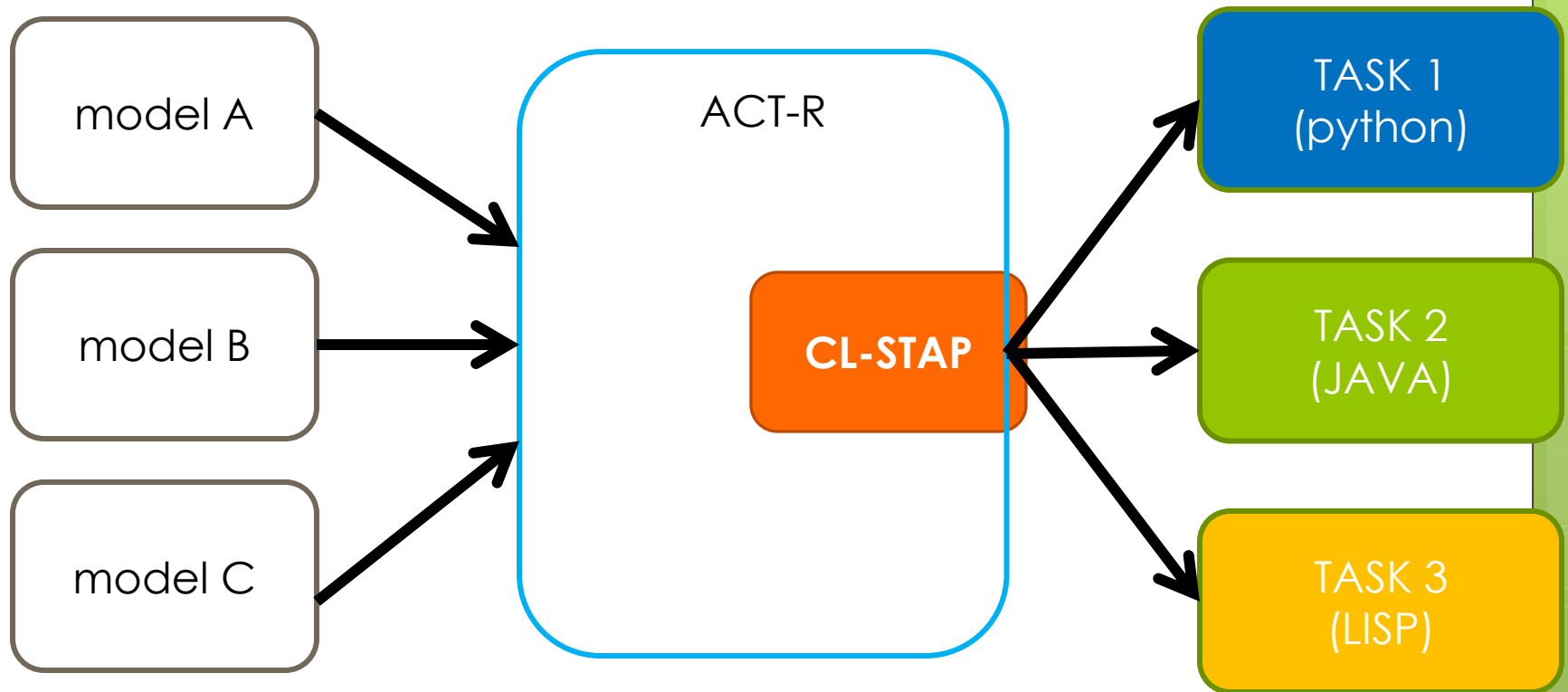
navigate



navigate



Serialized task-architecture interactions



Advancing the science

- Pace
 - experiments are easier to develop (**no GUI development**, API only)
 - simulations are easier to run (one-time cost to set up framework for API)
- Persistence
 - enable connection to multiple tasks
- Scale
 - separation of model/architecture from task enables scaling up of task and number of connected models (24 million?)

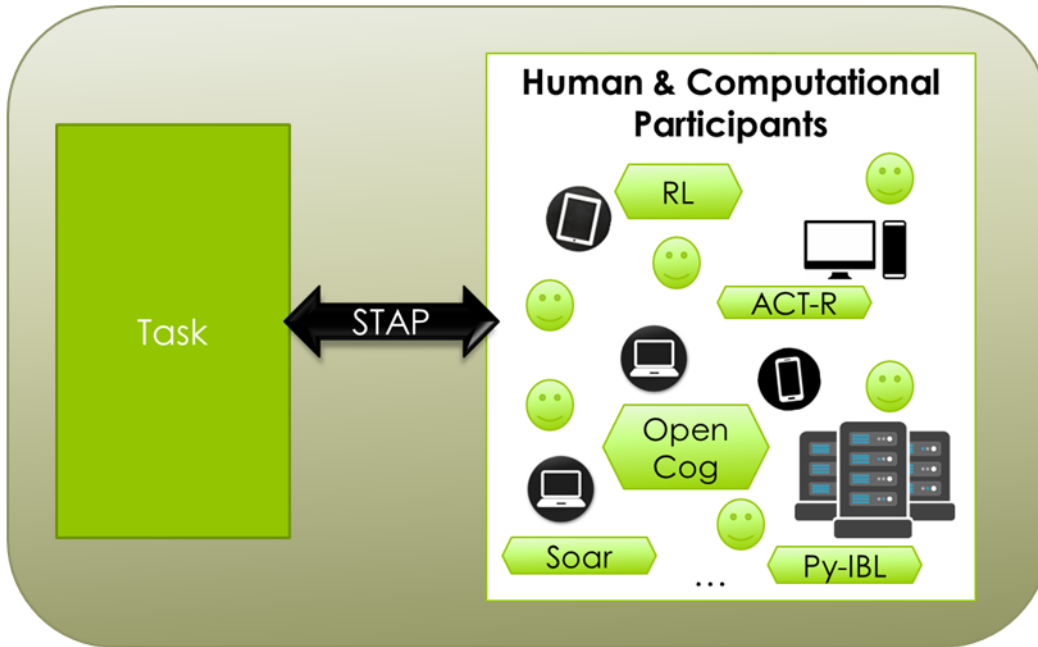
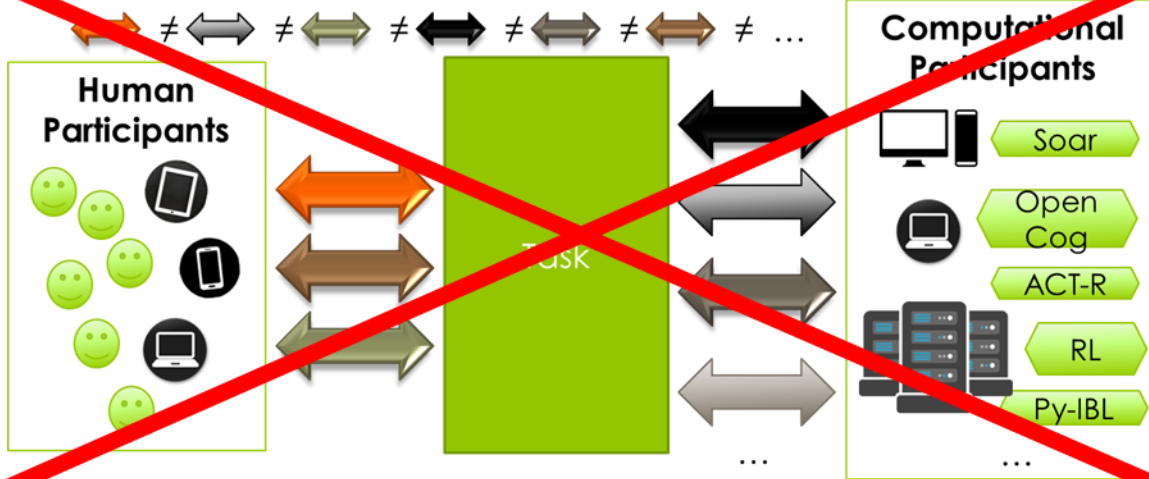
Computational Cognition Competitions

- Competitions and Grand Challenges
 - primary means to **motivate and galvanize** the research community to solve ambitious scientific and engineering challenges
- Developing a new task for a competition is not trivial
 - API, documentation, distribution/connection framework, human data
- No reuse of task interfacing between competitions
 - Increases engineering costs for participants
 - Reduces researcher buy-in

Other approaches

- web-apps (HTML5)
- visual (pixel-by-pixel)
- real-world (robotics)
- physical simulations (virtual world API's)
- task domain –specific API's (e.g. VGML)
- architecture-specific API's (e.g. JNI)

QUESTIONS?



<http://vdv7.github.io/stap>

