Rethinking Software Systems

A Friendly Introduction to

Behavioral Programming

Michael Bar-Sinai
@michbarsinai

Ben-Gurion University
of the Negev

Behavioral Programming (BP)

Programming Paradigm
Introduced in 2010 by David Harel, Assaf Marron and Gera Weiss

Rooted in Scenario-Based Programming (Damm, Harel, Marelly)

Large body of scientific work around this: algorithms, visualizations, event selection, interoperation.

Implementations in Java, C++, Erlang, StateCharts

DEVOXX 2018

first major non-academic conference

hosting a BP talk

DEVOXX 2018

first major non-academic conference

hosting a BP talk

Behavioral Programming with React
Luca Matteis, ReactjsDay, Italy, October 2018

Programming

Modeling

Programming

Modeling

Programming

Telling a computer

What to do

Modeling

Programming

Telling a computer

What to do

Modeling

Telling a computer

What could be done

Programming

Telling a computer

What to do

Modeling

Telling a computer

What could be done

can't be done

Programming

Telling a computer

What to do

Modeling

Telling a computer

What could be done

can't be done

Hello, world!

```
bp.registerBThread("bt-1", function(){
  bp.sync({request:bp.Event("hello")});
});

bp.registerBThread("bt-2", function(){
  bp.sync({request:bp.Event("world")});
});
```

Using BPjs, https://github.com/bThink-BGU/BPjs

Hello, world!

```
bp.registerBThread("bt-1", function(){
  bp.sync({request:bp.Event("hello")});
});

bp.registerBThread("bt-2", function(){
  bp.sync({request:bp.Event("world")});
});
```

Using BPjs, https://github.com/bThink-BGU/BPjs

Hello, world!

```
bp.registerBThread("bt-1", function(){
  bp.sync({request:bp.Event("hello")});
});

bp.registerBThread("bt-2", function(){
  bp.sync({request:bp.Event("world")});
});
```

Using BPjs, https://github.com/bThink-BGU/BPjs

Hello, world!

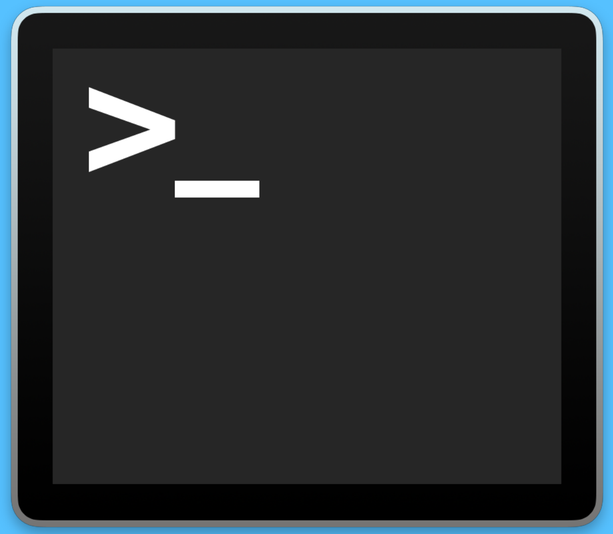
```
bp.registerBThread("bt-1", function(){
  bp.sync({request:bp.Event("hello")});
});

bp.registerBThread("bt-2", function(){
  bp.sync({request:bp.Event("world")});
});
```

Using BPjs, https://github.com/bThink-BGU/BPjs

Hello, world!

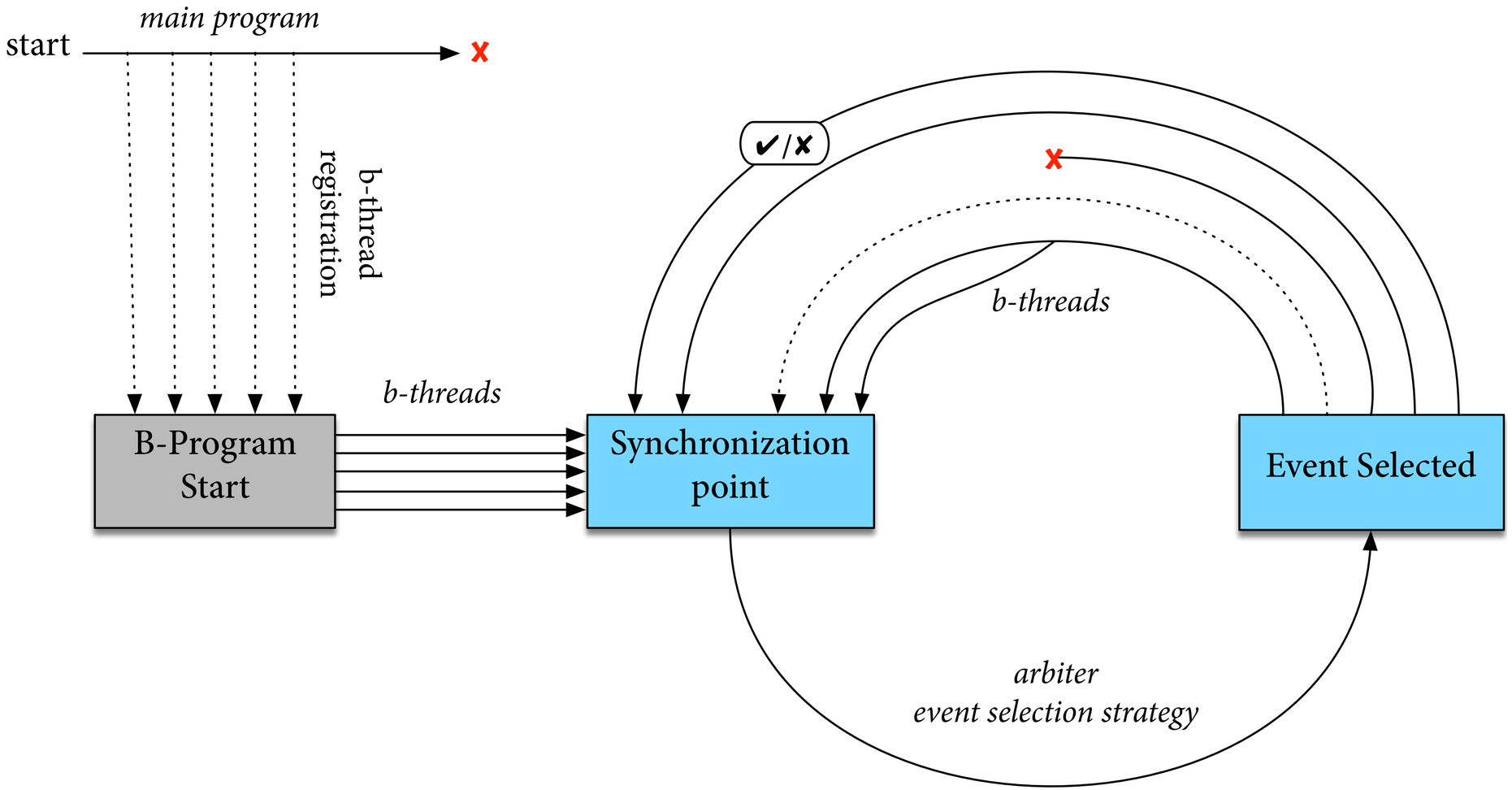
```
bp.registerBThread("bt-1", function(){
  bp.sync({request:bp.Event("hello")});
});

bp.registerBThread("bt-2", function(){
  bp.sync({request:bp.Event("world")});
});
```

Using BPjs, https://github.com/bThink-BGU/BPjs

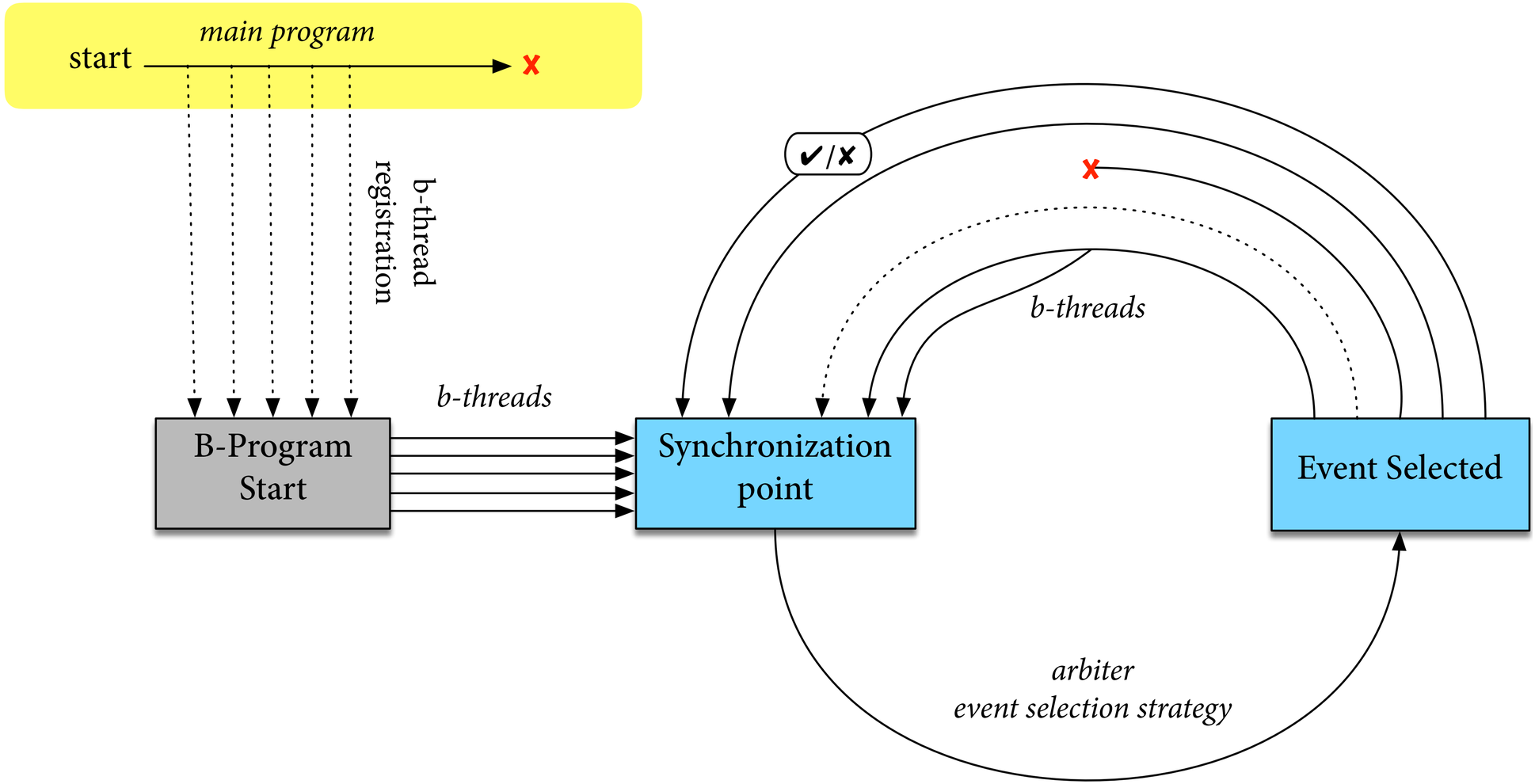


Demo

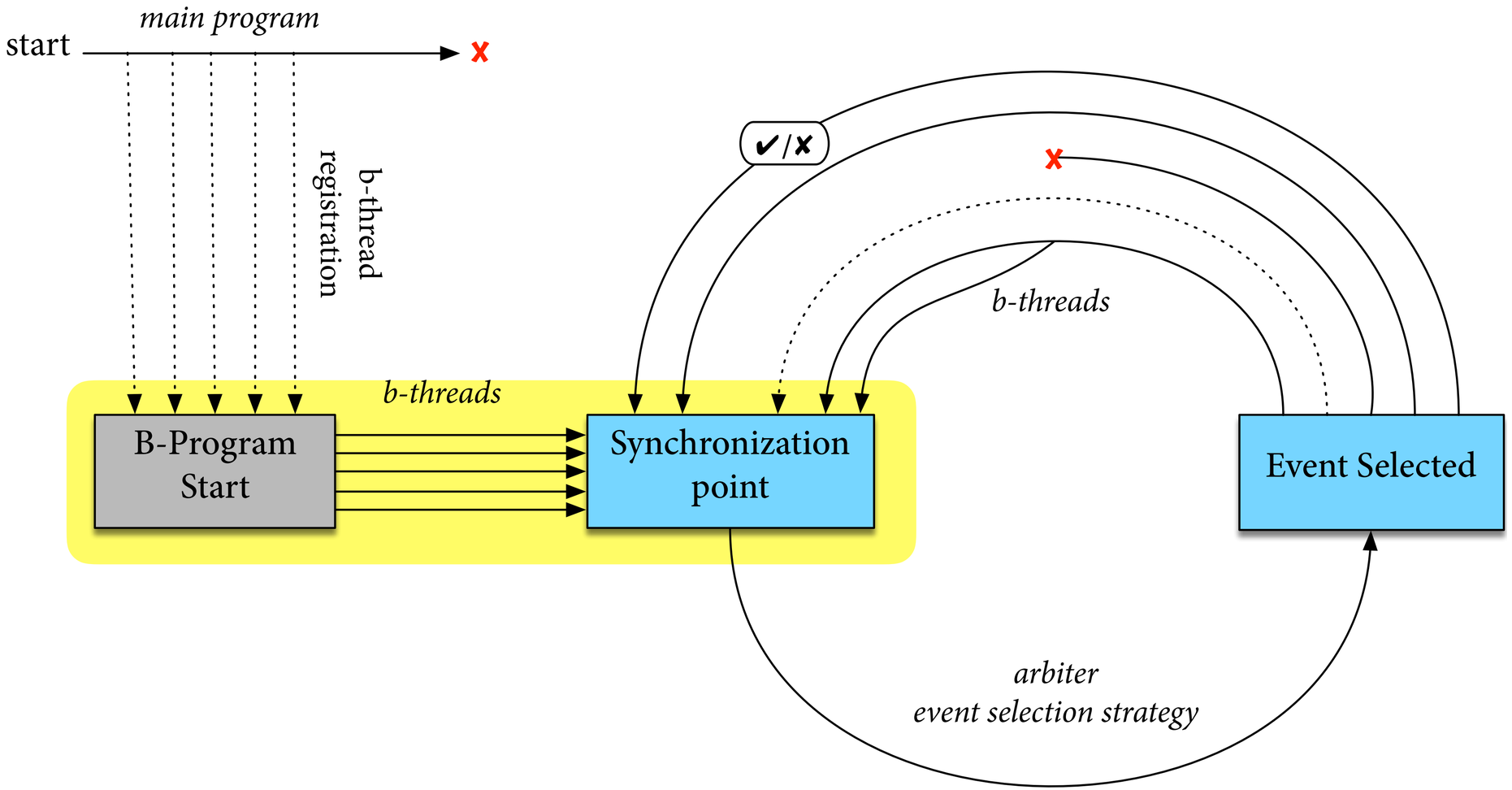
B-Program Life Cycle



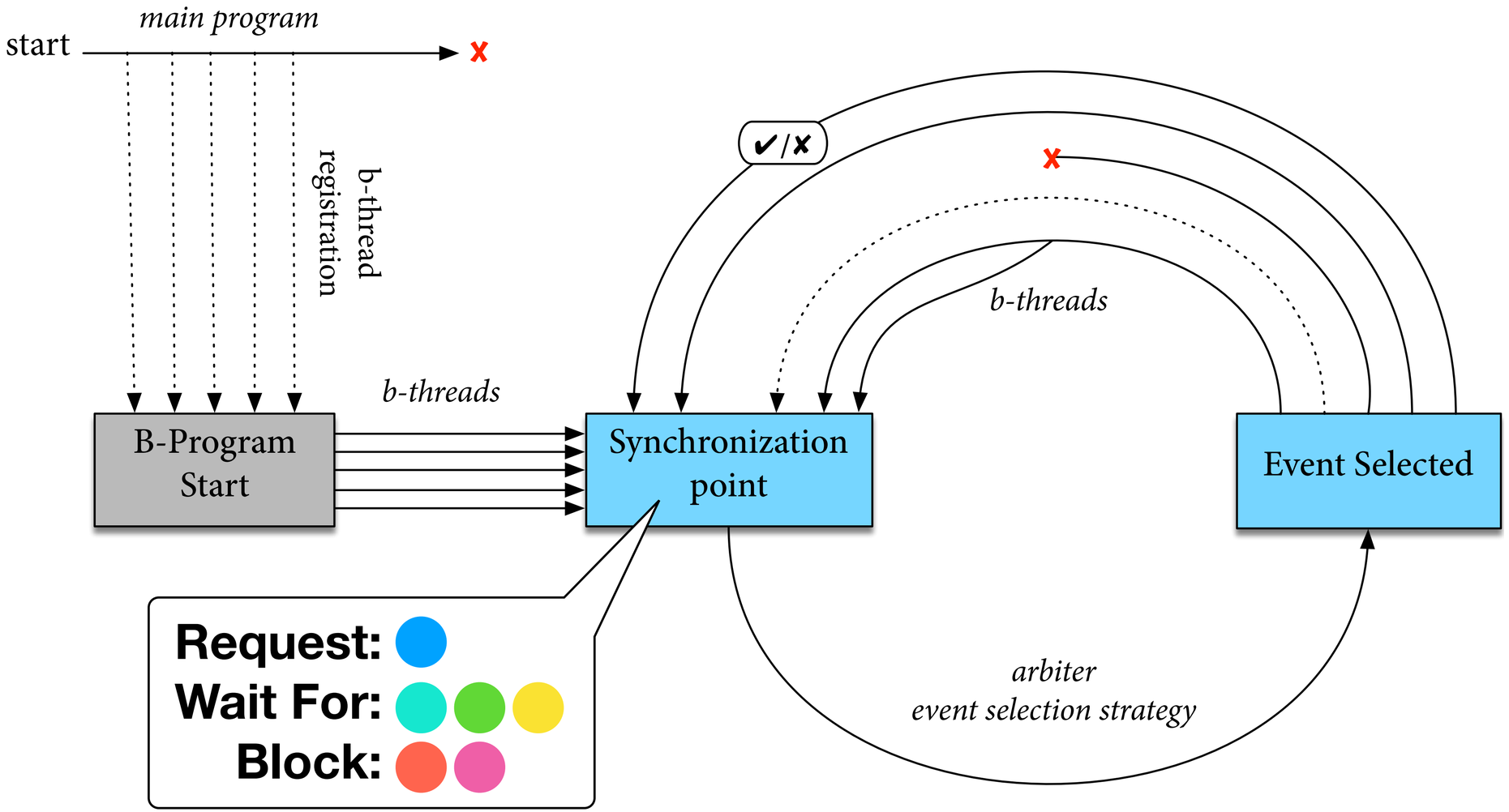
B-Program Life Cycle



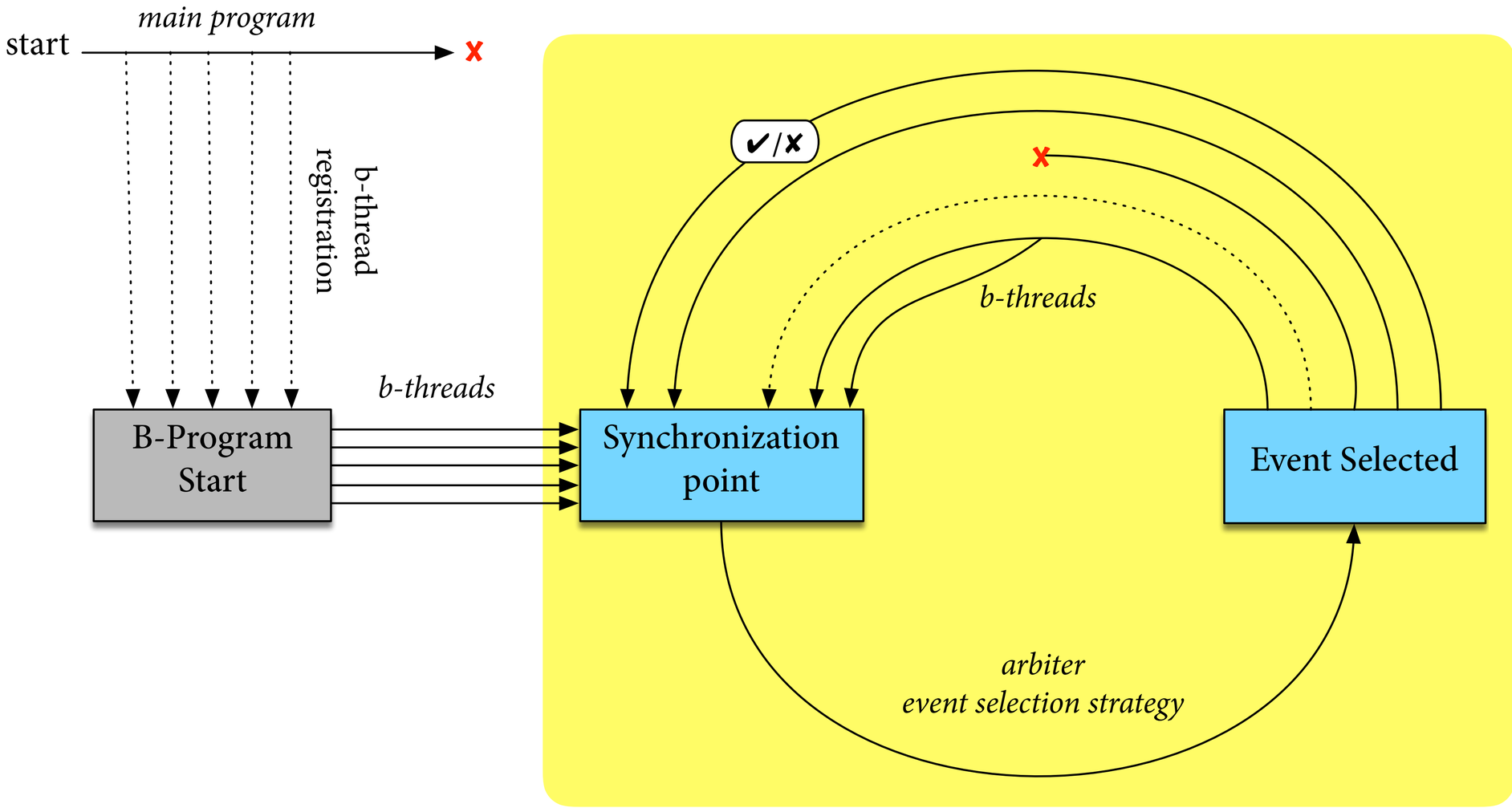
B-Program Life Cycle



B-Program Life Cycle



B-Program Life Cycle



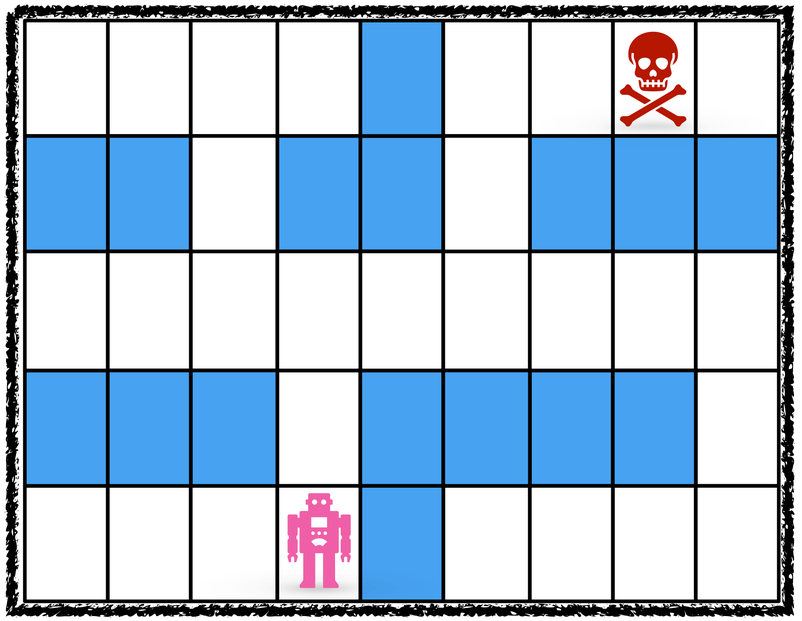
Demo

Hello World Takeaways

- Additivity / Composability
- Modularity
- Asynchronous-first approach

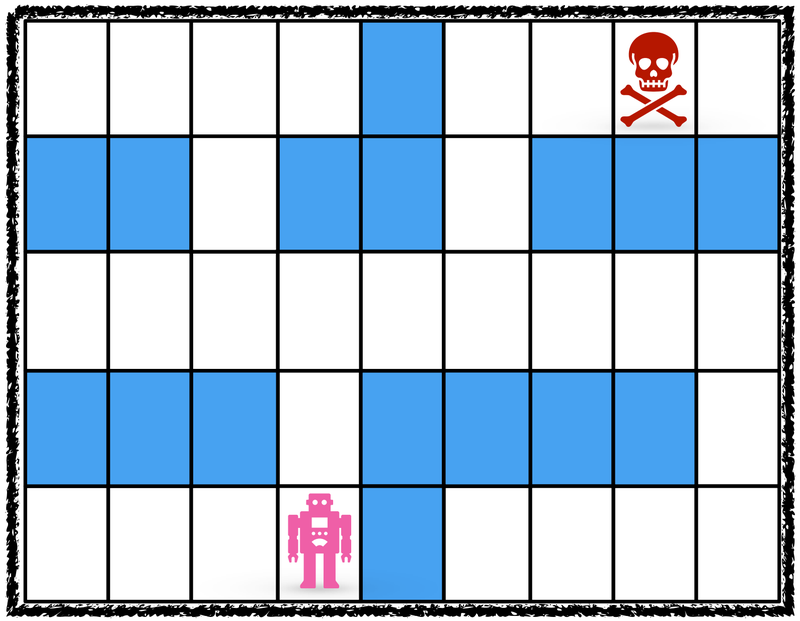
House Model

Simulate a robot moving inside a house



House Model

Simulate a robot moving inside a house



House Model

Simulate a robot moving inside a house

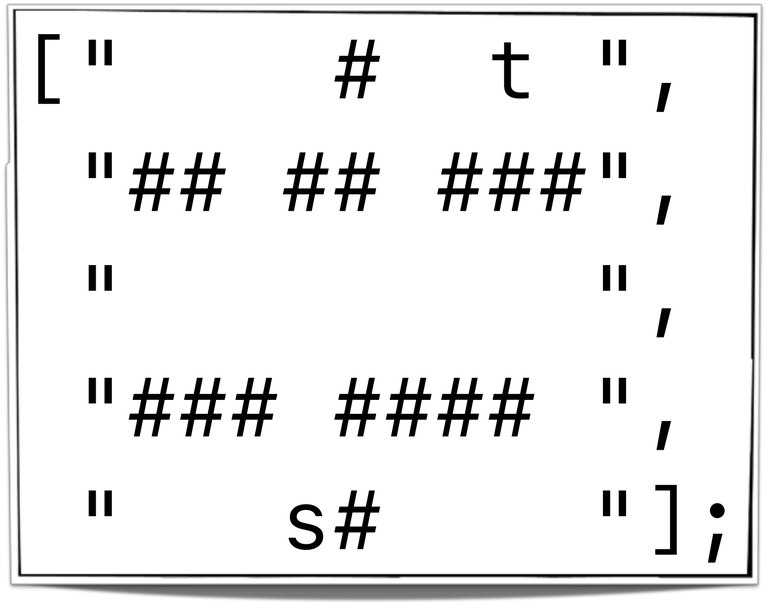
```
var house = [ "          #     t   ",
              " ## ##   ## ##   #### ",
              "                     ",
              " #### ##### ##### ## ",
              "        s #          " ];
```

Parsing into B-Threads: Idea

Parsing:

Iterate over characters in the house description string array.

Generate b-threads for each cell based on the character.



Parsing into B-Threads - Code

```
function parseMaze(mazeLines) {
    for ( var row=0; row<mazeLines.length; row++ ) {
        for ( var col=0; col<mazeLines[row].length; col++ ) {
            var currentPixel = mazeLines[row].substring(col,col+1);
            if ( currentPixel===" " ||
                 currentPixel==="t" ||
                 currentPixel==="s" ) {
                addSpaceCell(col, row);
                if ( currentPixel==="t" ) {
                    addTargetCell(col, row);
                } else if ( currentPixel==="s" ) {
                    addStartCell(col, row);
                }
            }
        }
    }
}
```

Parsing into B-Threads - Code

```
function parseMaze(mazeLines) {
    for ( var row=0; row<mazeLines.length; row++ ) {
        for ( var col=0; col<mazeLines[row].length; col++ ) {
            var currentPixel = mazeLines[row].substring(col,col+1);
            if ( currentPixel===" " ||
                 currentPixel==="t" ||
                 currentPixel==="s" ) {
                addSpaceCell(col, row);
                if ( currentPixel==="t" ) {
                    addTargetCell(col, row);
                } else if ( currentPixel==="s" ) {
                    addStartCell(col, row);
                }
            }
        }
    }
}
```

Parsing into B-Threads - Code

```
function parseMaze(mazeLines) {
    for ( var row=0; row<mazeLines.length; row++ ) {
        for ( var col=0; col<mazeLines[row].length; col++ ) {
            var currentPixel = mazeLines[row].substring(col,col+1);
            if ( currentPixel===" " ||
                 currentPixel==="t" ||
                 currentPixel==="s" ) {
                addSpaceCell(col, row);
                if ( currentPixel==="t" ) {
                    addTargetCell(col, row);
                } else if ( currentPixel==="s" ) {
                    addStartCell(col, row);
                }
            }
        }
    }
}
```

Parsing into B-Threads - Code

```
function parseMaze(mazeLines) {
    for ( var row=0; row<mazeLines.length; row++ ) {
        for ( var col=0; col<mazeLines[row].length; col++ ) {
            var currentPixel = mazeLines[row].substring(col,col+1);
            if ( currentPixel===" " ||
                 currentPixel==="t" ||
                 currentPixel==="s" ) {
                addSpaceCell(col, row);
                if ( currentPixel==="t" ) {
                    addTargetCell(col, row);
                } else if ( currentPixel==="s" ) {
                    addStartCell(col, row);
                }
            }
        }
    }
}
```

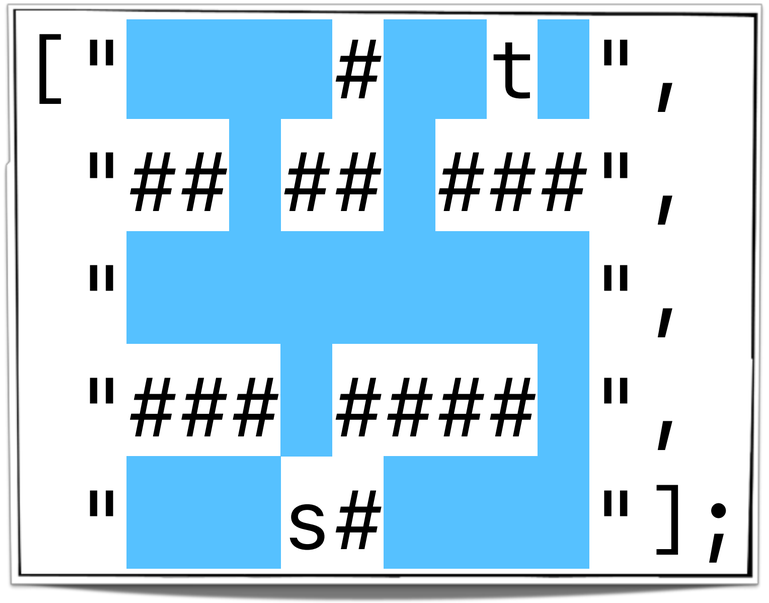
Parsing into B-Threads - Code

```
function parseMaze(mazeLines) {
    for ( var row=0; row<mazeLines.length; row++ ) {
        for ( var col=0; col<mazeLines[row].length; col++ ) {
            var currentPixel = mazeLines[row].substring(col,col+1);
            if ( currentPixel===" " ||
                 currentPixel==="t" ||
                 currentPixel==="s" ) {
                addSpaceCell(col, row);
                if ( currentPixel==="t" ) {
                    addTargetCell(col, row);
                } else if ( currentPixel==="s" ) {
                    addStartCell(col, row);
                }
            }
        }
    }
}
```

"Different types of cells have different behaviors"

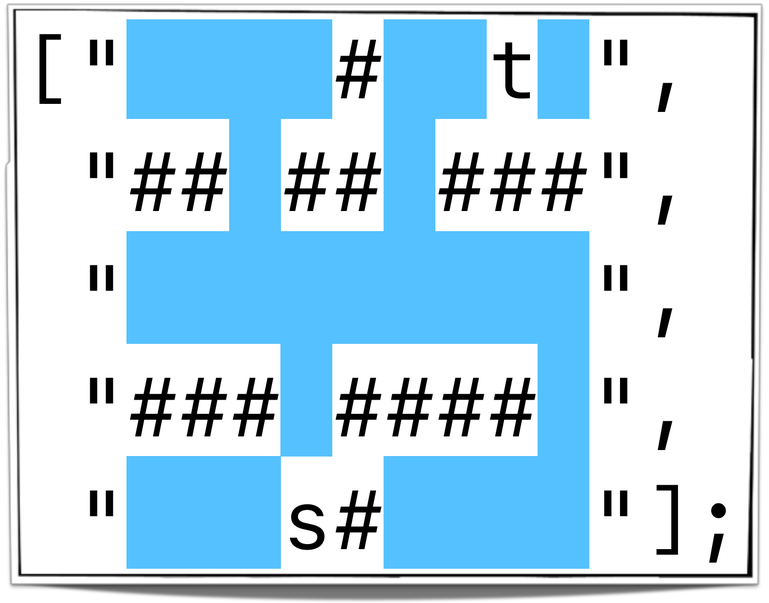
Behaviors for a **Space Cell**

```
function addSpaceCell( col, row ) {
    bp.registerBThread("cell(c:"+col+" r:"+row+")",
        function() {
            while ( true ) {
                bp.sync({waitFor:adjacentCellEntries(col, row)});
                bp.sync({
                    request: enterEvent(col, row),
                    waitFor: anyEntrance
                });
            }
        }
    );
}
```



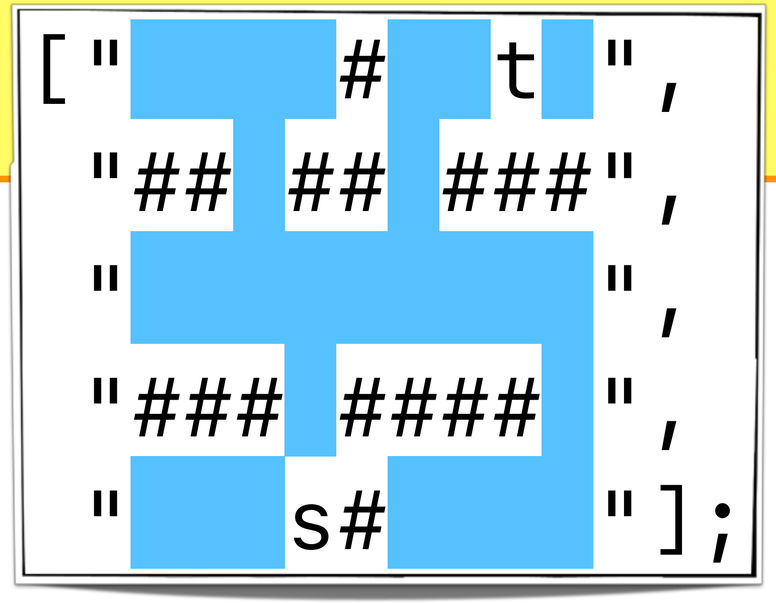
Behaviors for a **Space Cell**

```
function addSpaceCell( col, row ) {
    bp.registerBThread("cell(c:"+col+" r:"+row+")",
        function() {
            while ( true ) {
                bp.sync({waitFor:adjacentCellEntries(col, row)});
                bp.sync({
                    request: enterEvent(col, row),
                    waitFor: anyEntrance
                });
            }
        }
    );
}
```



Behaviors for a **Space Cell**

```
function addSpaceCell( col, row ) {
    bp.registerBThread("cell(c:"+col+" r:"+row+")",
        function() {
            while ( true ) {
                bp.sync({waitFor:adjacentCellEntries(col, row)});
                bp.sync({
                    request: enterEvent(col, row),
                    waitFor: anyEntrance
                });
            }
        }
    );
}
```



Behaviors for a **Target Cell**

```
function addSpaceCell( col, row ) {...}
```

+

```
function addTargetCell(col, row) {
    bp.registerBThread("Target(c:"+col+" r:"+row+")", function(){
        bp.sync({
            waitFor: enterEvent(col, row)
        });

        bp.sync({
            request: TARGET_FOUND,
            block: bp.allExcept( TARGET_FOUND )
        });
    });
}
```

```
["      #   t  ",
 "## ## ###",
 "              ",
 "### #### ",
 "    s#      "];
```

Behaviors for a **Target Cell**

```
function addSpaceCell( col, row ) {...}
```

+

```
function addTargetCell(col, row) {
    bp.registerBThread("Target(c:"+col+" r:"+row+")", function(){
        bp.sync({
            waitFor: enterEvent(col, row)
        });

        bp.sync({
            request: TARGET_FOUND,
            block: bp.allExcept( TARGET_FOUND )
        });
    });
}
```

```
["      #   t  ",
 "## ## ###",
 "              ",
 "### #### ",
 "    s#      "];
```

Behaviors for a **Target Cell**

```
function addSpaceCell( col, row ) {...}
```

+

```
function addTargetCell(col, row) {
    bp.registerBThread("Target(c:"+col+" r:"+row+")", function(){
        bp.sync({
            waitFor: enterEvent(col, row)
        });

        bp.sync({
            request: TARGET_FOUND,
            block: bp.allExcept( TARGET_FOUND )
        });
    });
}
```

```
["      #   t  ",
 "## ## ###",
 "              ",
 "### #### ",
 "   s#      "];
```

Behaviors for a **Target Cell**

```
function addSpaceCell( col, row ) {...}
```

+

```
function addTargetCell(col, row) {
    bp.registerBThread("Target(c:"+col+" r:"+row+")", function(){
        bp.sync({
            waitFor: enterEvent(col, row)
        });

        bp.sync({
            request: TARGET_FOUND,
            block: bp.allExcept( TARGET_FOUND )
        });
    });
}
```

```
["      #   t  ",
 "## ## ###",
 "              ",
 "### #### ",
 "    s#      "];
```

Behaviors for a **Start Cell**

```
function addSpaceCell( col, row ) {...}
```

+

```
function addStartCell(col, row) {
    bp.registerBThread("starter(c:"+col+" r:"+row+")", function() {
        bp.sync({
            request:enterEvent(col,row)
        });
    });
}
```

```
[" #  t ",
 "## ## ###",
 "          ",
 "### #### ",
 "  s#     "];
```

Behaviors for a **Start Cell**

```
function addSpaceCell( col, row ) {...}
```

+

```
function addStartCell(col, row) {
    bp.registerBThread("starter(c:"+col+" r:"+row+")", function() {
        bp.sync({
            request:enterEvent(col,row)
        });
    });
}
```

```
[" #  t ",
 "## ## ###",
 "          ",
 "### #### ",
 "   s#     "];
```

Behaviors for a **Start Cell**

```
function addSpaceCell( col, row ) {...}
```

+

```
function addStartCell(col, row) {
    bp.registerBThread("starter(c:"+col+" r:"+row+")", function() {
        bp.sync({
            request:enterEvent(col,row)
        });
    });
}
```

```
[" #  t ",
 "## ## ###",
 "          ",
 "### #### ",
 "   s#     "];
```

Behaviors for a **Start Cell**

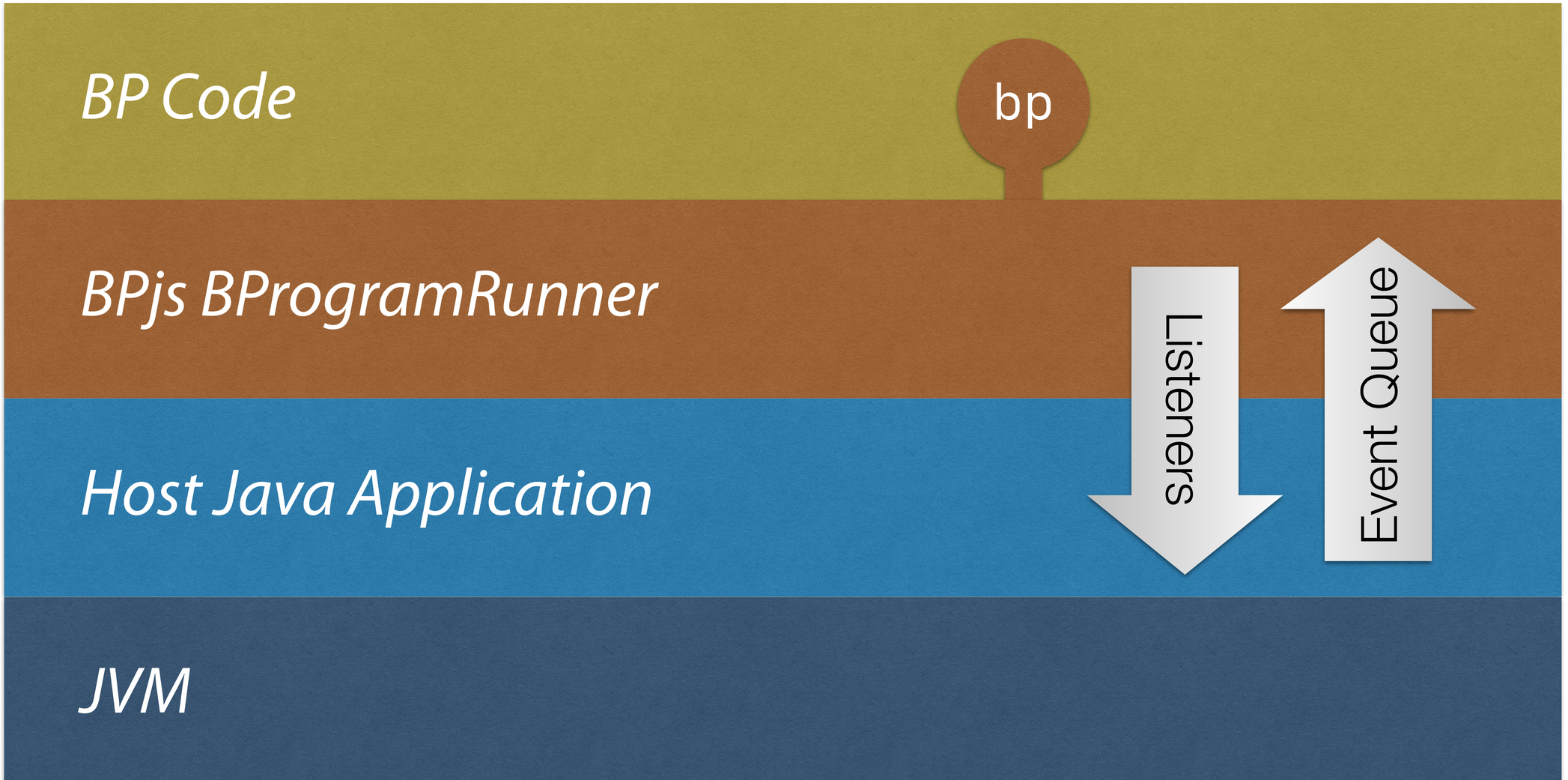
```
function addSpaceCell( col, row ) {...}
```

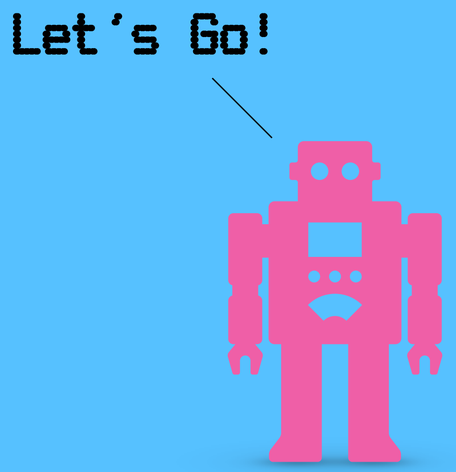
+

```
function addStartCell(col, row) {
    bp.registerBThread("starter(c:"+col+" r:"+row+")", function() {
        bp.sync({
            request:enterEvent(col,row)
        });
    });
}
```

```
[" #  t ",
 "## ## ###",
 "          ",
 "### #### ",
 "  s#     "];
```

Runtime Stack





Demo

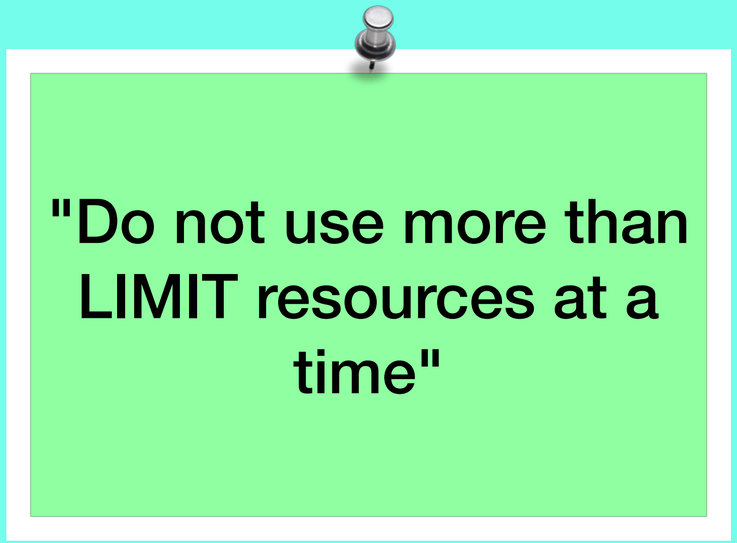
"After the first three pages, clean the ink jets"

"After the first three pages, clean the ink jets"

```
bp.registerBThread("cleaner", function(){
      bp.sync({waitFor:PrintPage});
      bp.sync({waitFor:PrintPage});
      bp.sync({waitFor:PrintPage});
      bp.sync({
          request: CleanInkJets,
            block: PrintPage
      });
});
```

"Every three pages, clean the ink jets"

```
bp.registerBThread("cleaner", function(){
    while ( true ) {
        bp.sync({waitFor:PrintPage});
        bp.sync({waitFor:PrintPage});
        bp.sync({waitFor:PrintPage});
        bp.sync({
            request: CleanInkJets,
              block: PrintPage
        });
    }
});
```


"Do not use more than LIMIT resources at a time"

"Do not use more than LIMIT resources at a time"

```
bp.registerBThread("limitConsumption", function(){
    var count = 0;
    while (true) {
        var evt = bp.sync({waitFor:[AnyUsageStart, AnyUsageEnd]});
        if ( AnyUsageStart.contains(evt) ) {
            count++;
        } else if ( AnyUsageEnd.contains(evt) ) {
            count--;
        }
        if ( count === LIMIT ) {
            bp.log.info("limiting...")
            bp.registerBThread("tempBlock", function(){
                bp.sync({
                    waitFor:AnyUsageEnd,
                    block:AnyUsageStart
                });
            });
        }
    }
});
```

"Do not use more than LIMIT resources at a time"

```
bp.registerBThread("limitConsumption", function(){
    var count = 0;
    while (true) {
        var evt = bp.sync({waitFor:[AnyUsageStart, AnyUsageEnd]});
        if ( AnyUsageStart.contains(evt) ) {
            count++;
        } else if ( AnyUsageEnd.contains(evt) ) {
            count--;
        }
        if ( count === LIMIT ) {
            bp.log.info("limiting...")
            bp.registerBThread("tempBlock", function(){
                bp.sync({
                    waitFor:AnyUsageEnd,
                    block:AnyUsageStart
                });
            });
        }
    }
});
```

"when we run out of a products *p1* and *p2*, do not allow new orders until the stock is renewed"

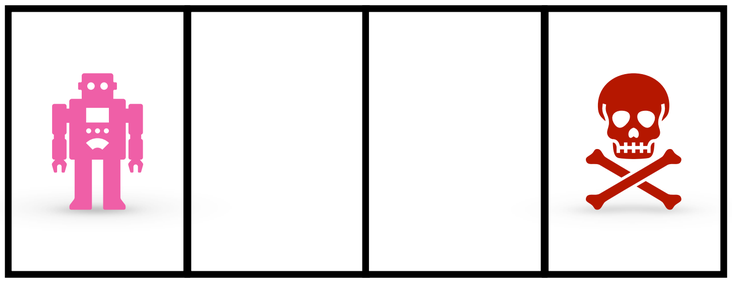
"when we run out of a products *p1* and *p2*, do not allow new orders until the stock is renewed"

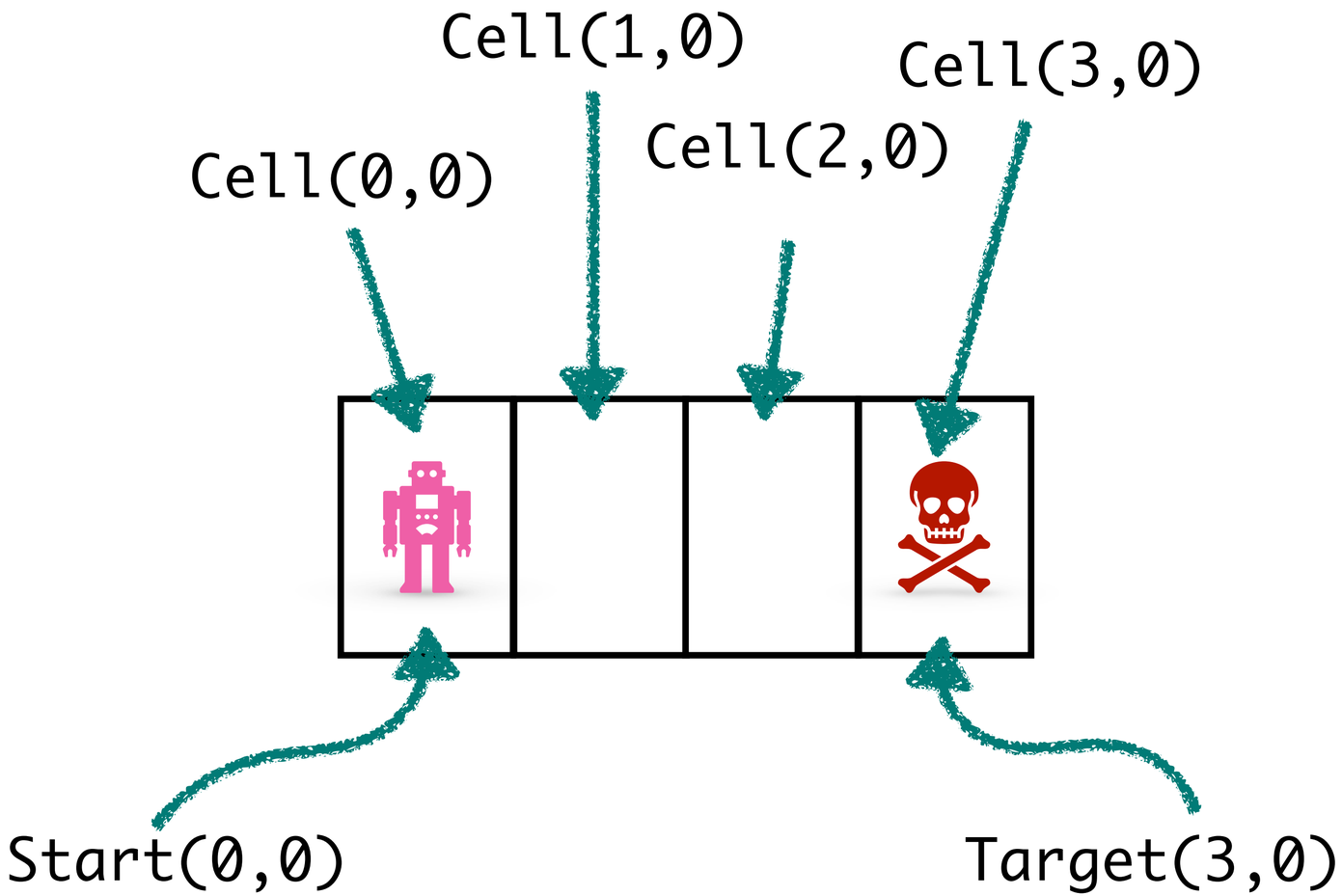
```
function blockOrdersWhenOutOf( p ) {
    return function(){
        while(true) {
            bp.sync({waitFor: OutOfStock(p)});
            bp.sync({
                waitFor: RefillDone(p),
                block: Order(p)
            });
        };
    };
}
bp.registerBThread(blockOrdersWhenOutOf(p1));
bp.registerBThread(blockOrdersWhenOutOf(p2));
```

Requirement Alignment

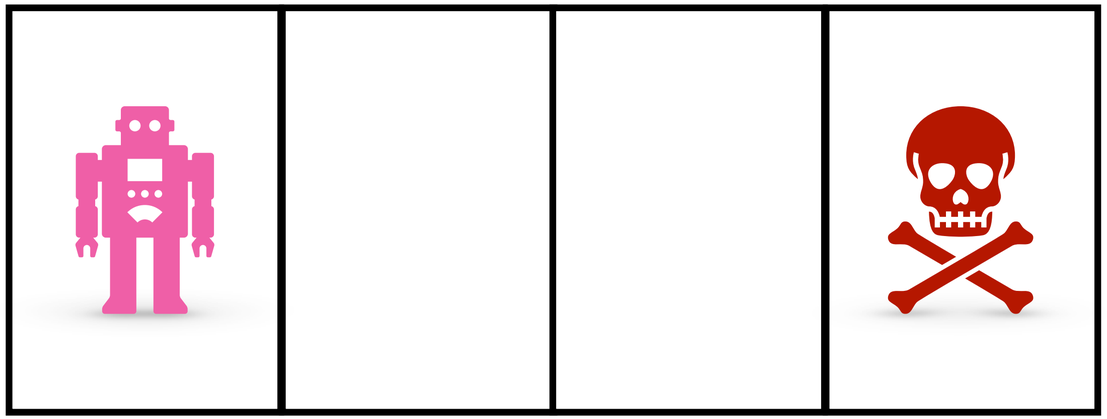
"when we run out of a products *p1* and *p2*, do not allow new orders until the stock is renewed"

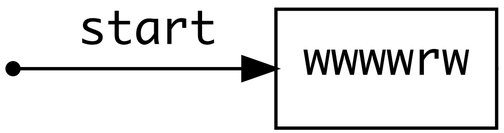
```
function blockOrdersWhenOutOf( p ) {
    return function(){
        while(true) {
            bp.sync({waitFor: OutOfStock(p)});
            bp.sync({
                waitFor: RefillDone(p),
                block: Order(p)
            });
        };
    };
}
bp.registerBThread(blockOrdersWhenOutOf(p1));
bp.registerBThread(blockOrdersWhenOutOf(p2));
```



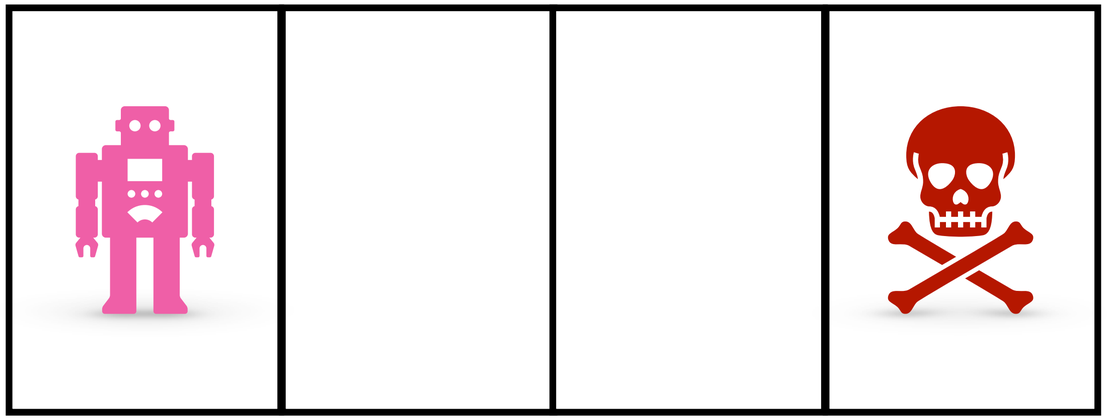

Cell(1,0)
Cell(3,0)
Cell(0,0)
Cell(2,0)
Start(0,0)
Target(3,0)

Target(3,0)
Start(0,0)
Cell(3,0)
Cell(2,0)
Cell(1,0)
Cell(0,0)

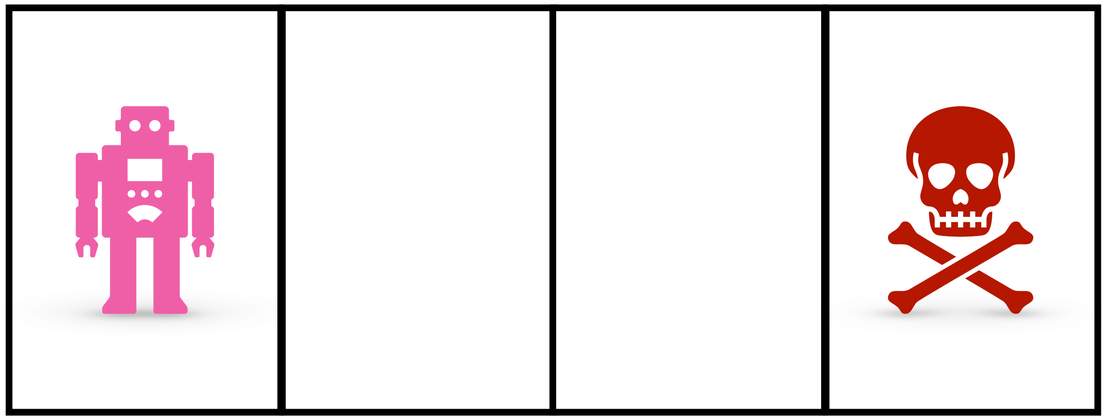


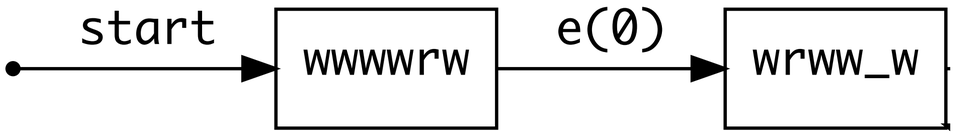


Target(3,0)
Start(0,0)
Cell(3,0)
Cell(2,0)
Cell(1,0)
Cell(0,0)

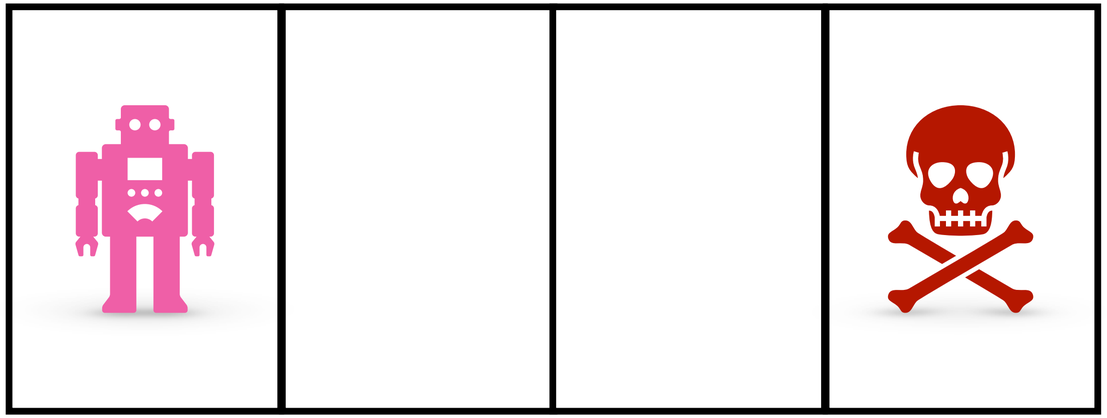


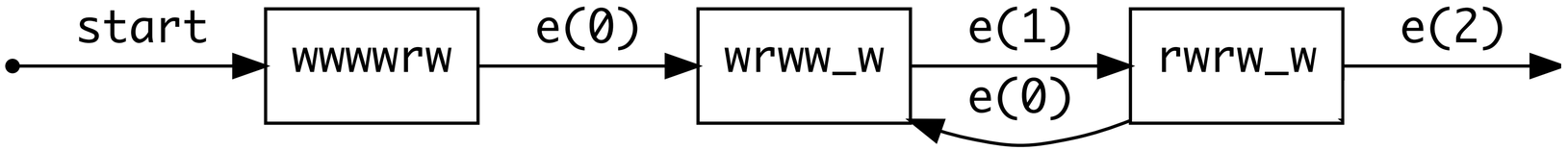
Target(3,0)
Start(0,0)
Cell(3,0)
Cell(2,0)
Cell(1,0)
Cell(0,0)



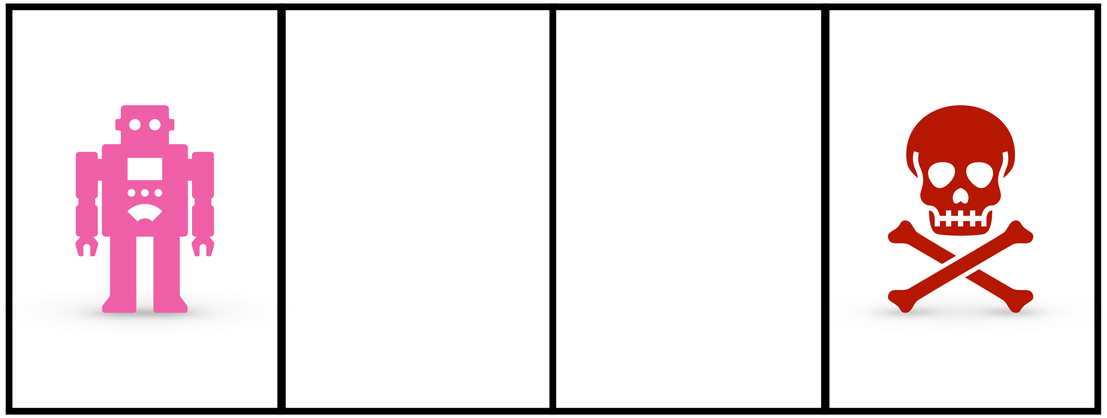


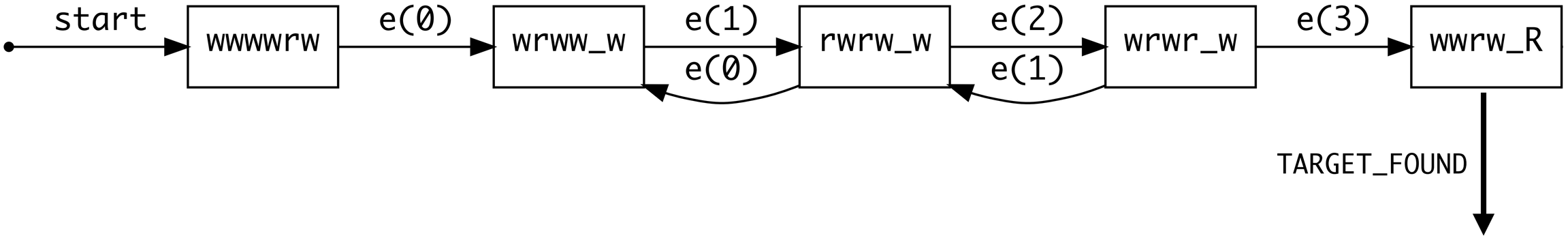
Target(3,0)
Start(0,0)
Cell(3,0)
Cell(2,0)
Cell(1,0)
Cell(0,0)



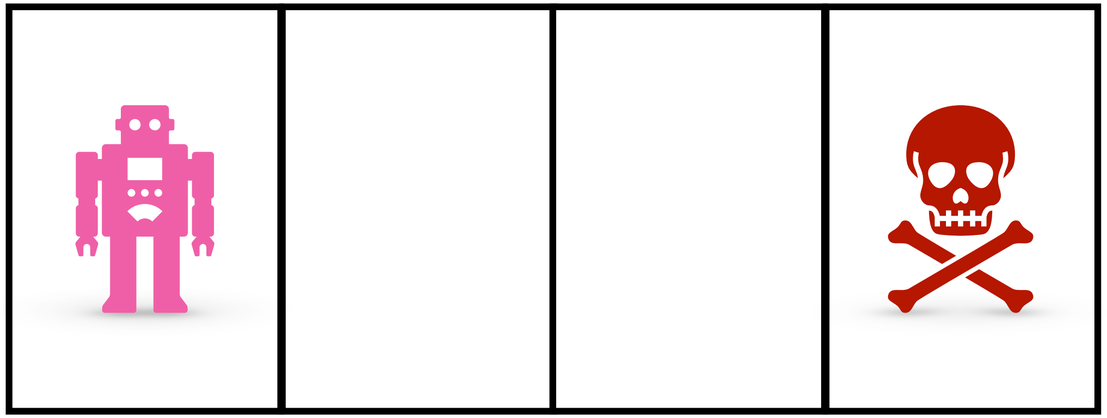


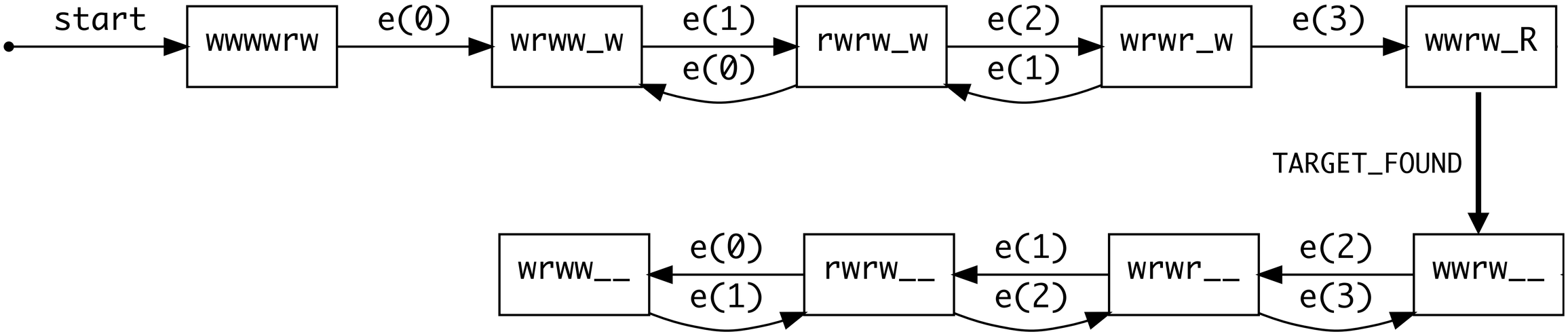
Target(3,0)
Start(0,0)
Cell(3,0)
Cell(2,0)
Cell(1,0)
Cell(0,0)



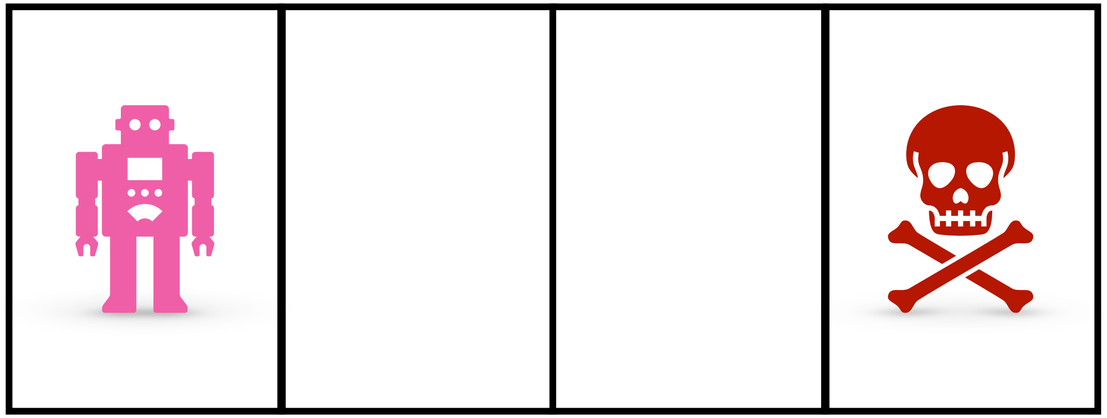


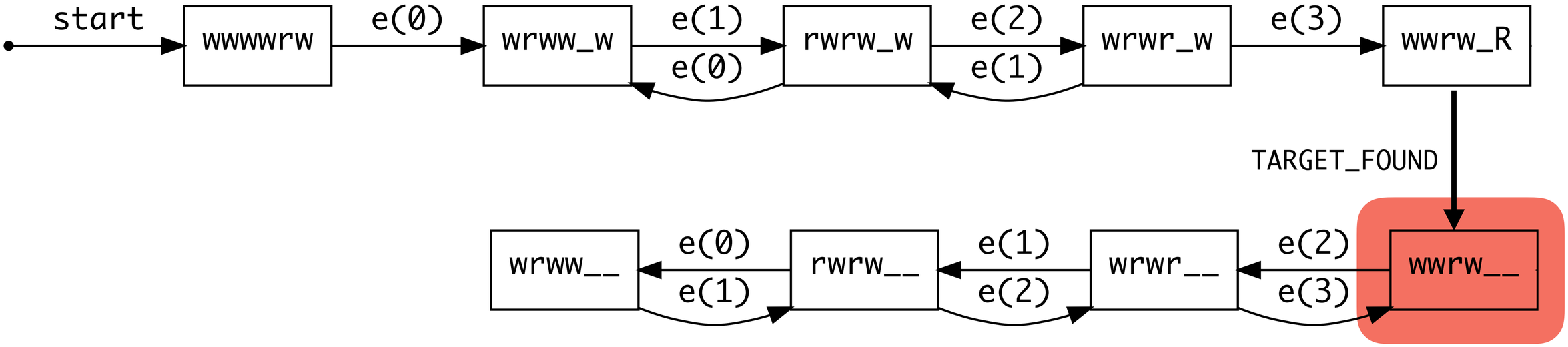
Target(3,0)
Start(0,0)
Cell(3,0)
Cell(2,0)
Cell(1,0)
Cell(0,0)





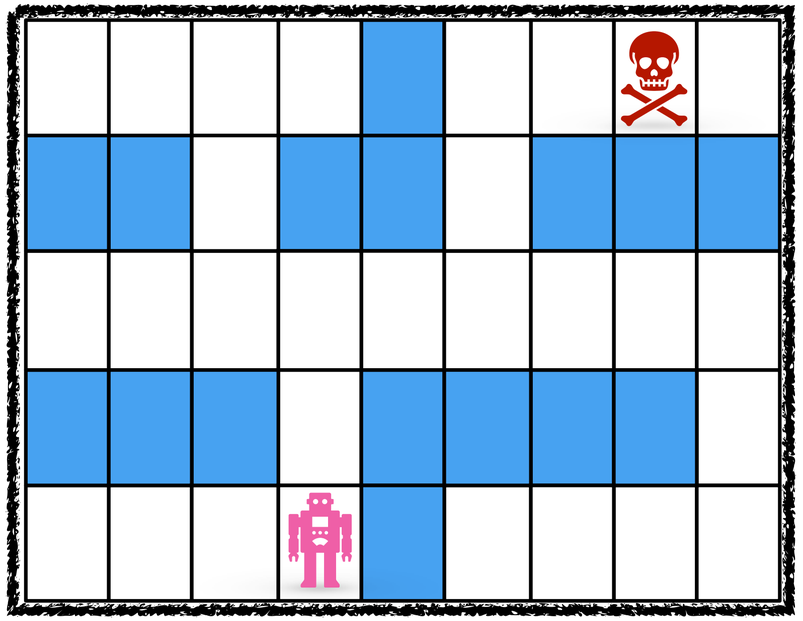
Target(3,0)
Start(0,0)
Cell(3,0)
Cell(2,0)
Cell(1,0)
Cell(0,0)





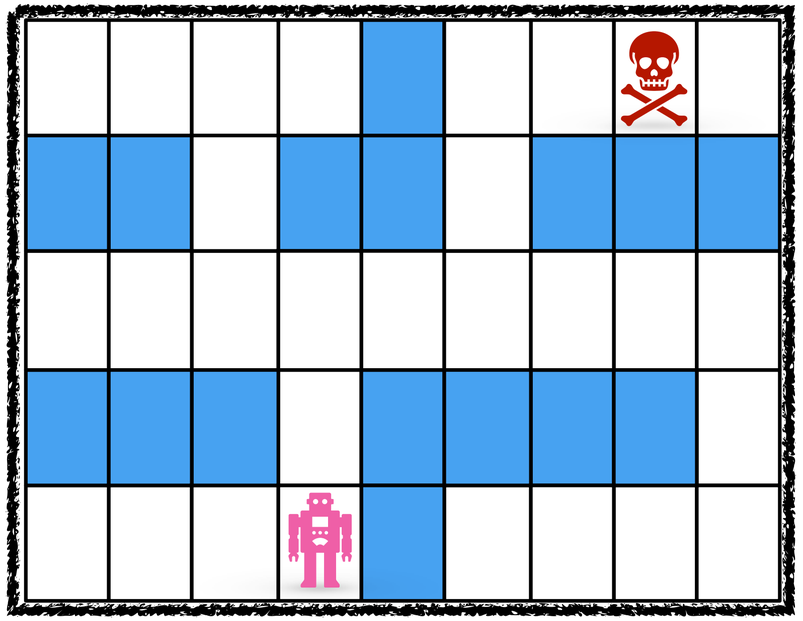
Verification Detecting "Bad States"

```
bp.registerBThread("robot-falling", function(){
    bp.sync({waitFor:TARGET_FOUND});
    bp.ASSERT(false, "Robot fell into trap");
});
```



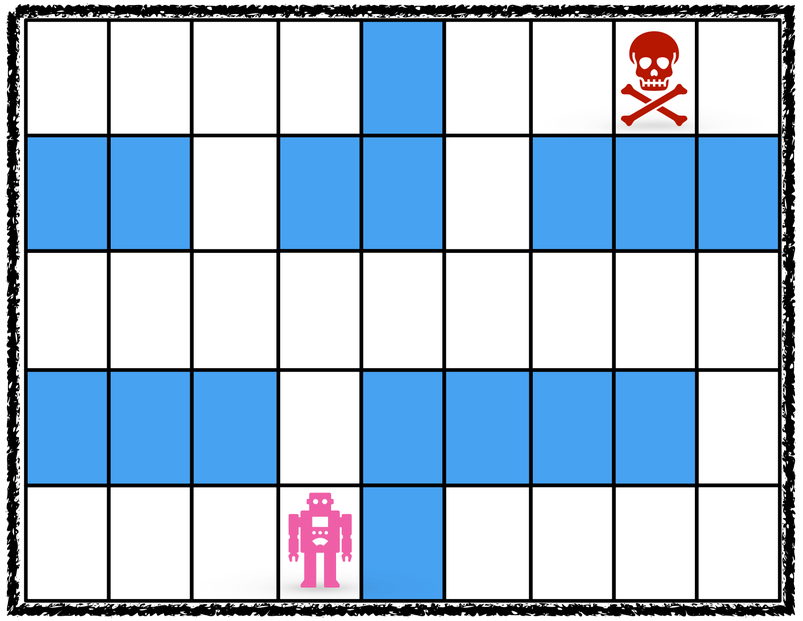
Verification Detecting "Bad States"

```
bp.registerBThread("robot-falling", function(){
    bp.sync({waitFor:TARGET_FOUND});
    bp.ASSERT(false, "Robot fell into trap");
});
```



Verification Detecting "Bad States"

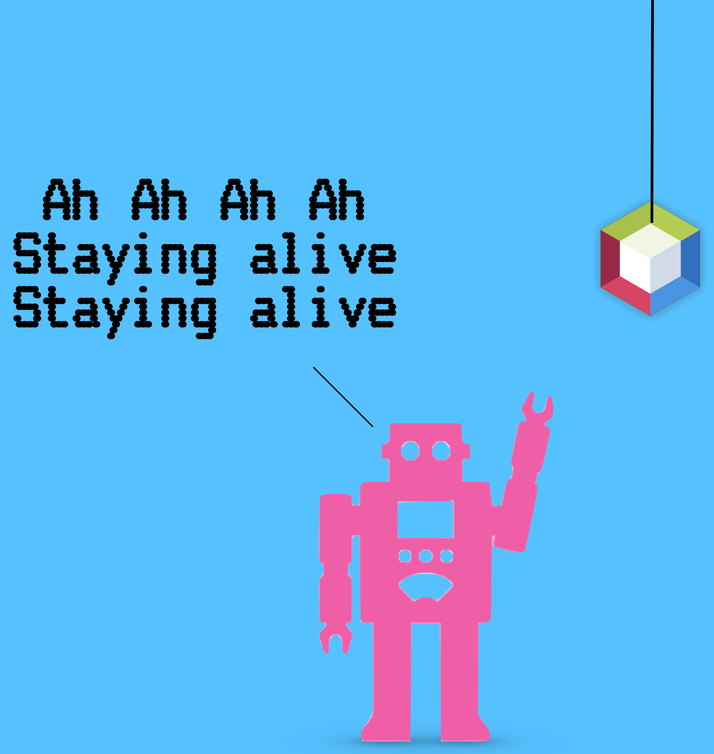
```
bp.registerBThread("robot-falling", function(){
    bp.sync({waitFor:TARGET_FOUND});
    bp.ASSERT(false, "Robot fell into trap");
});
```



Keeping the Robot Safe

1. Add the "robot-falling" b-thread to the maze b-program
2. Traverse state graph (currently using DFS, A* demonstrated)
3. When hitting failed assertion, output trace
4. Use assumptions where possible (right)

```
bp.registerBThread("onlyOnce",
 function(){
    var visited = [];
    while (true) {
      var evt = bp.sync({
        waitFor: anyEntrance,
          block: visited
      });
      visited.push(evt);
    }
});
```



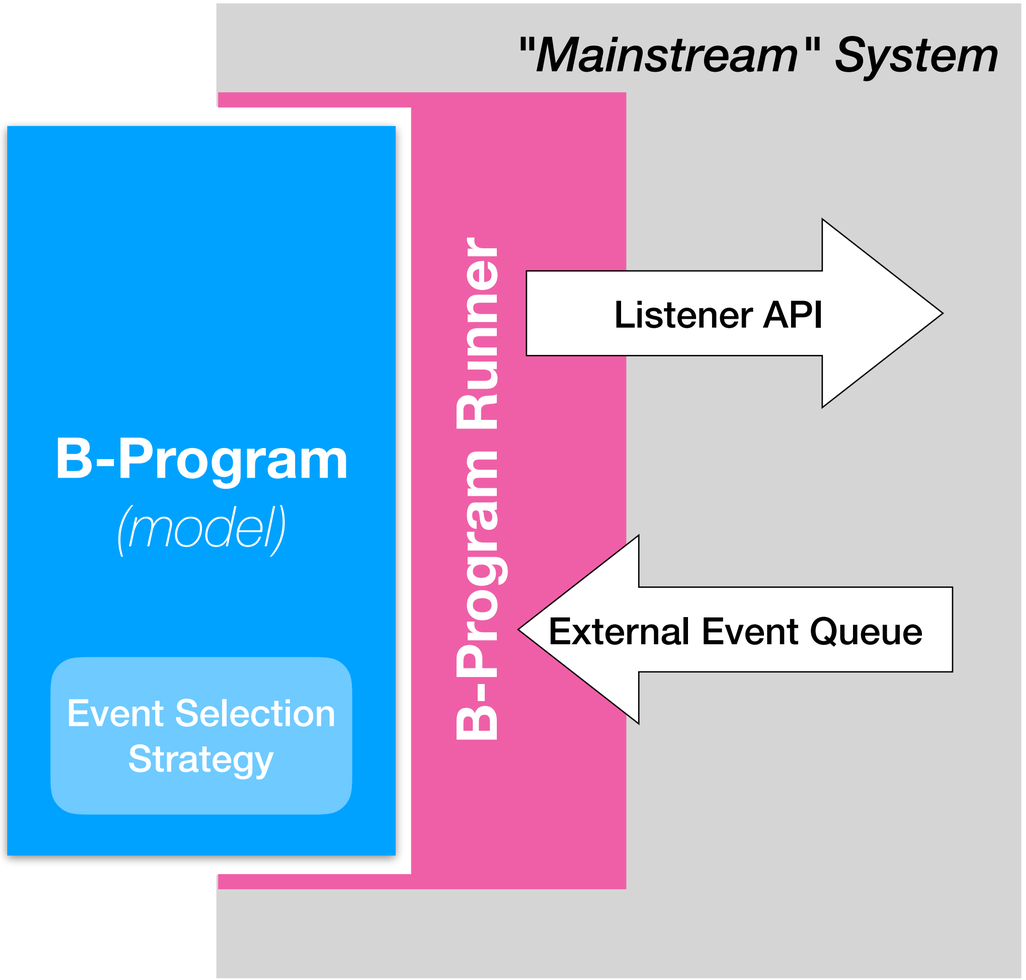
Demo

Verification in BPjs

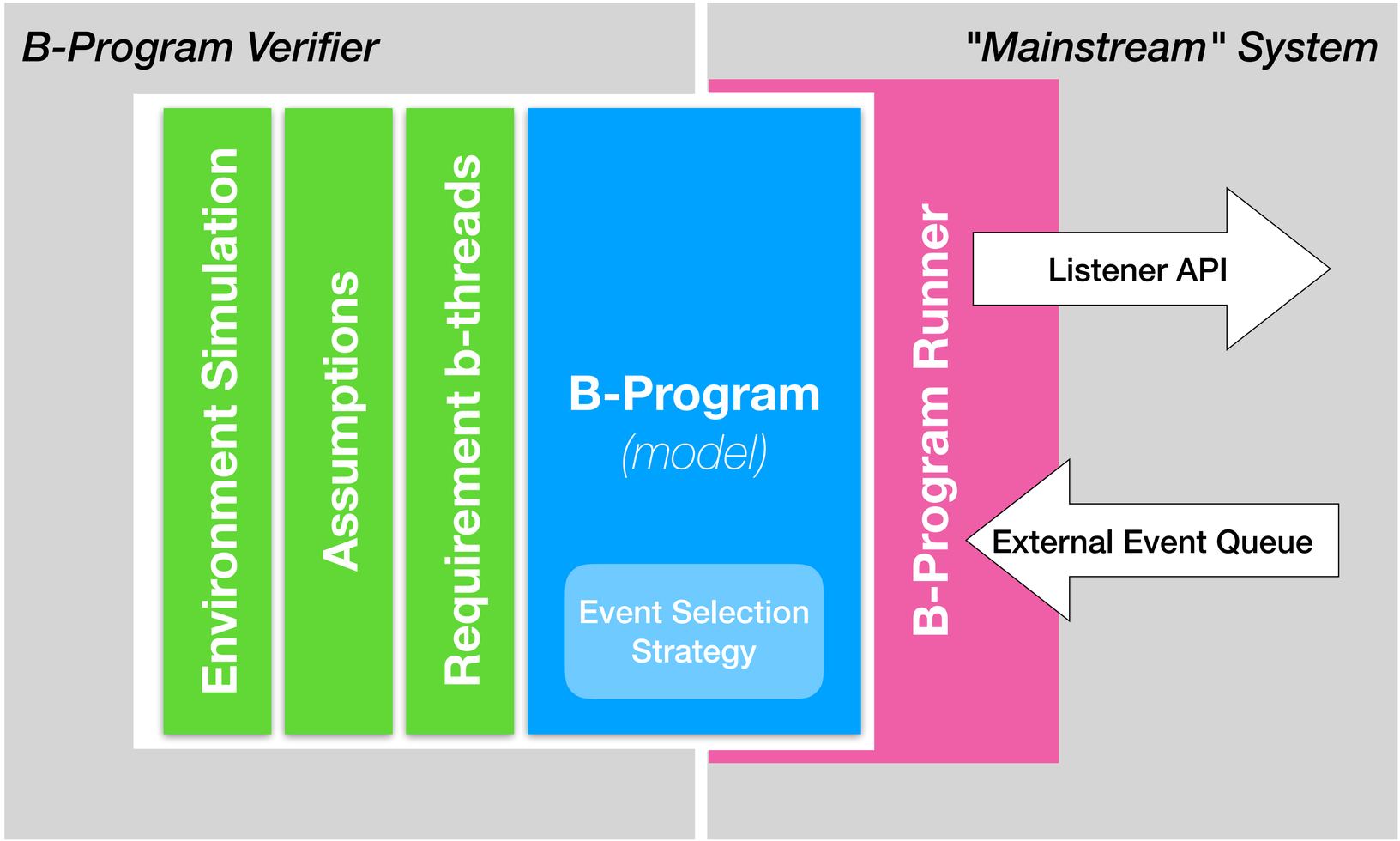
```
BProgram bprog = new StringBProgram(mazesSource);
bprog.prependSource(mazeJs);
bprog.appendSource(assumptionsSrc);
bprog.appendSource(requirementSrc);
DfsBProgramVerifier vfr = new DfsBProgramVerifier();
vfr.setDetectDeadlocks(false);
vfr.setProgressListener(...);
VerificationResult res = vfr.verify(bprog);
```

SE ? ❤️ BP

SE ? ❤ BP


"Mainstream" System
B-Program
(model)
Event Selection Strategy
B-Program Runner
Listener API
External Event Queue

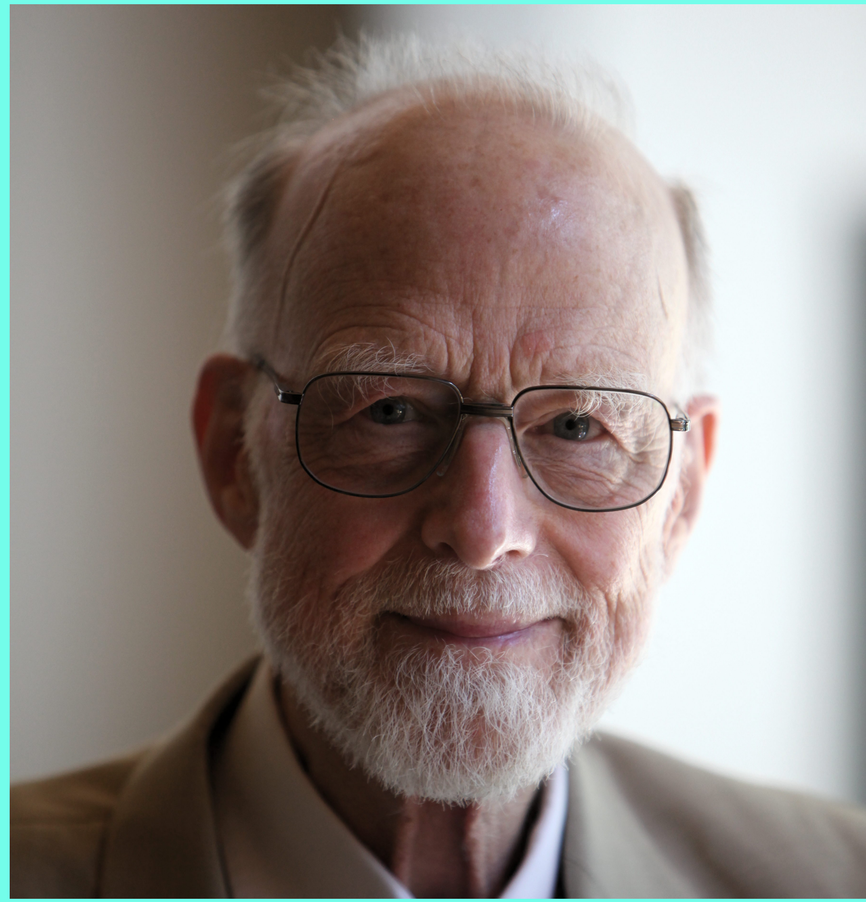
SE ? ❤ BP


B-Program Verifier
"Mainstream" System
Environment Simulation
Assumptions
Requirement b-threads
B-Program
(model)
Event Selection Strategy
B-Program Runner
Listener API
External Event Queue

So What?

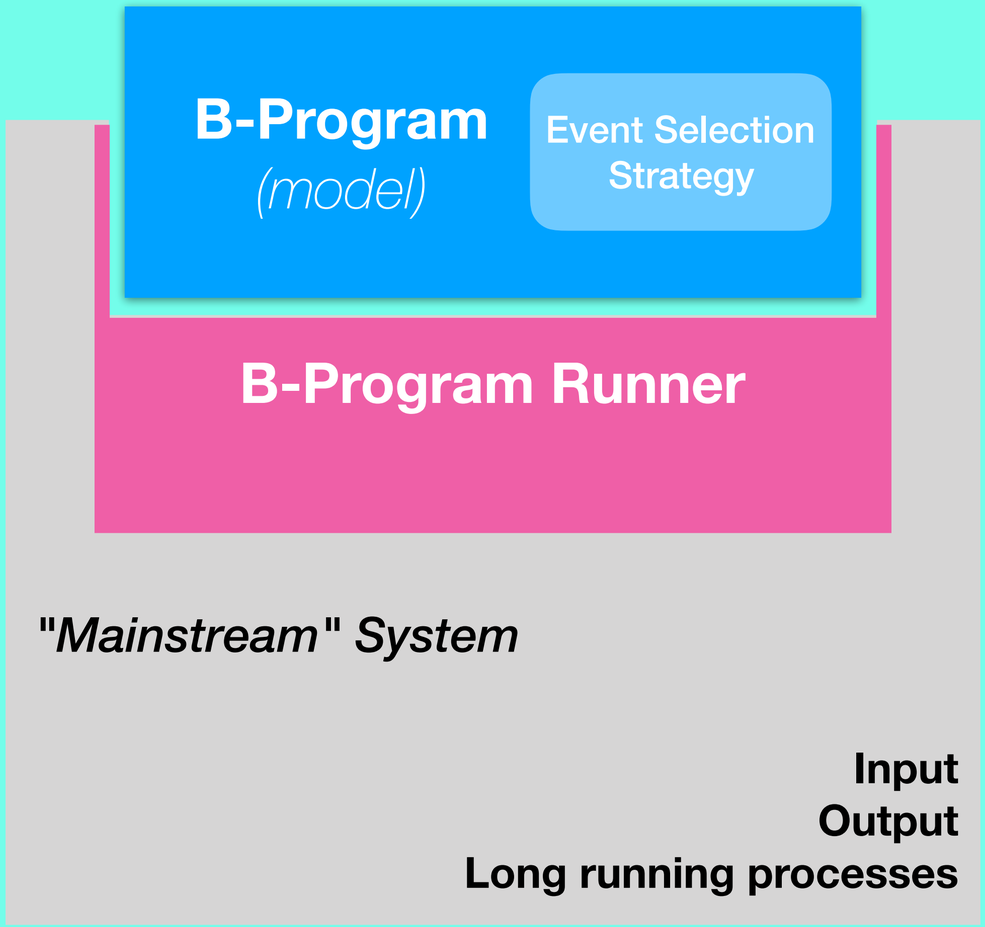
There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies.

C.A.R Hoare, in
The Emperor's Old Clothes,
Turing Award lecture, 1980

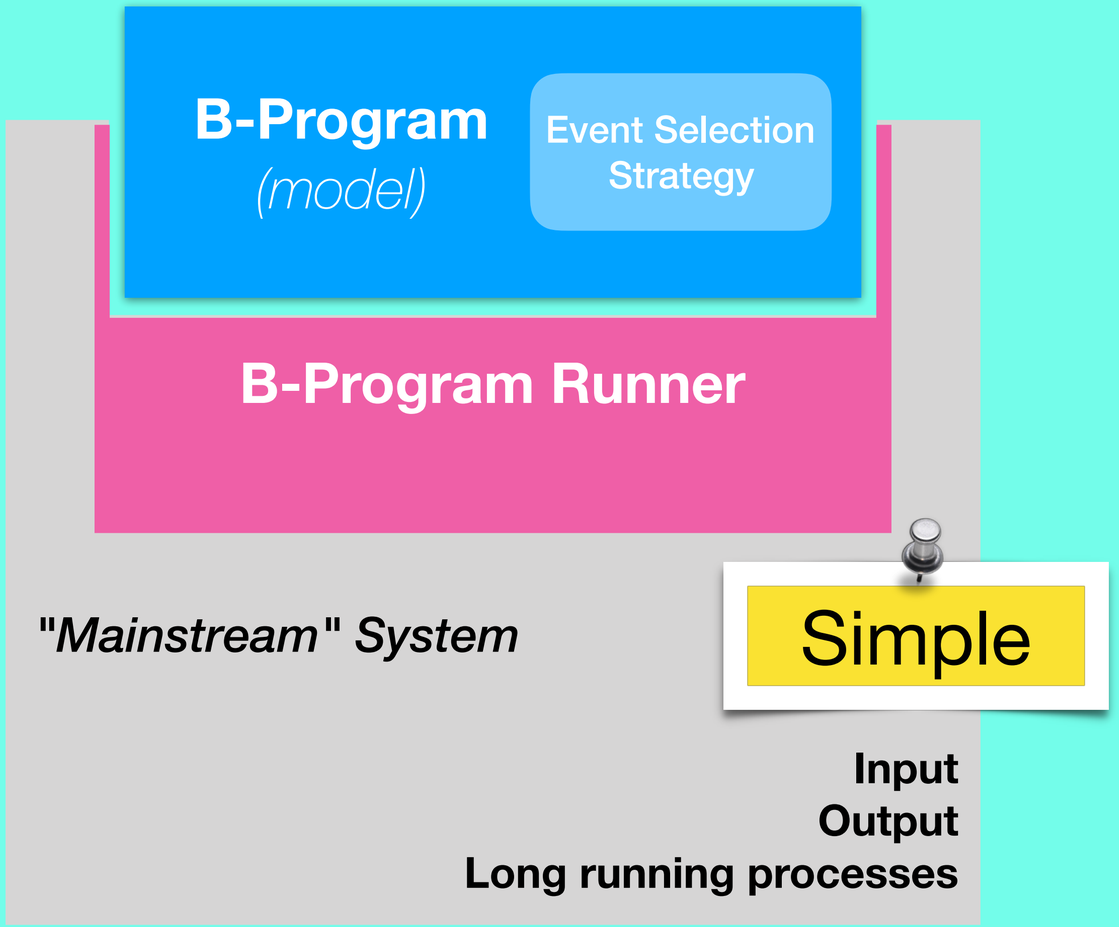


Photograph by Rama, Wikimedia Commons, Cc-by-sa-2.0-fr

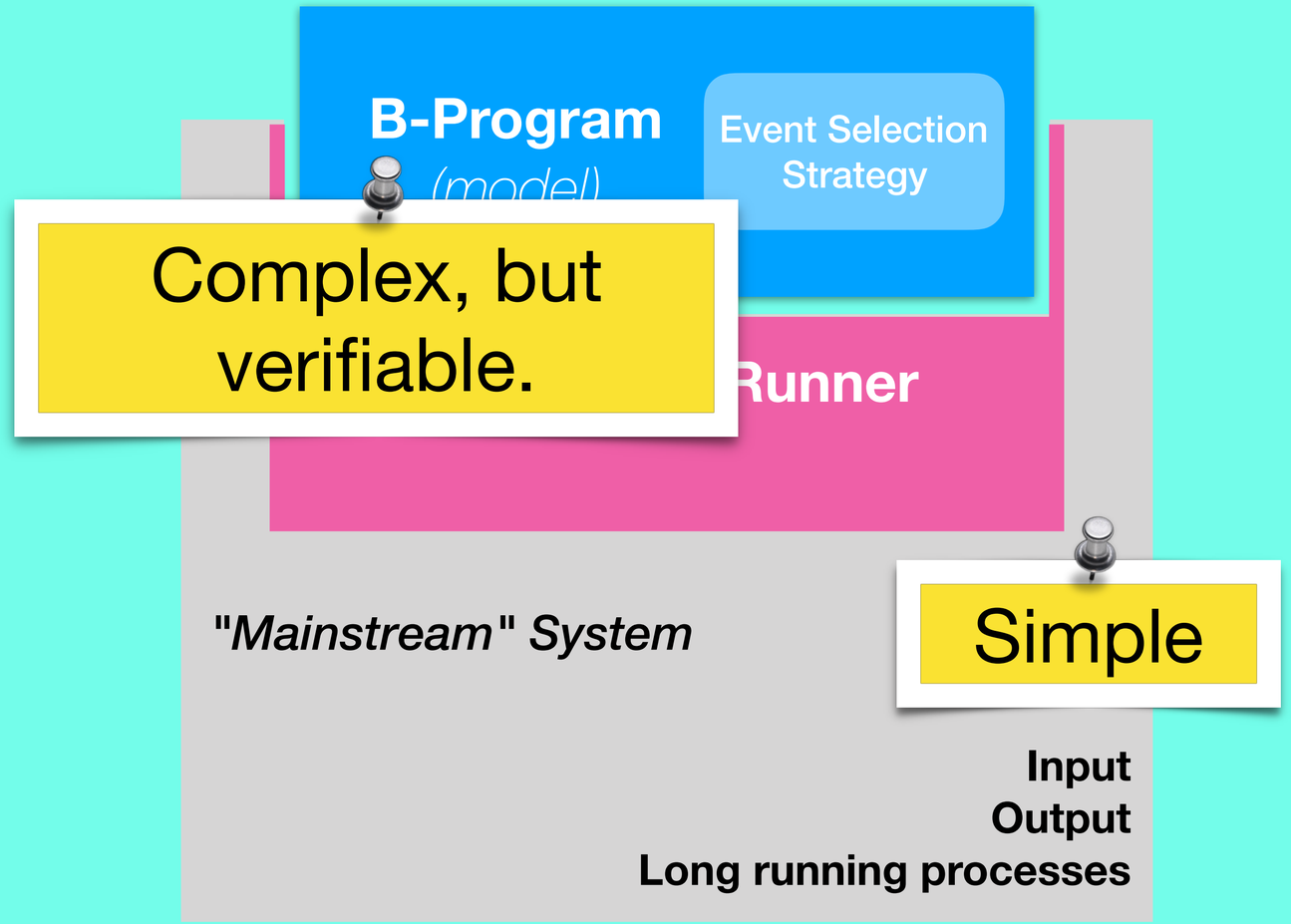
Why not Both?


B-Program
(model)
Event Selection Strategy
B-Program Runner
"Mainstream" System
Input
Output
Long running processes

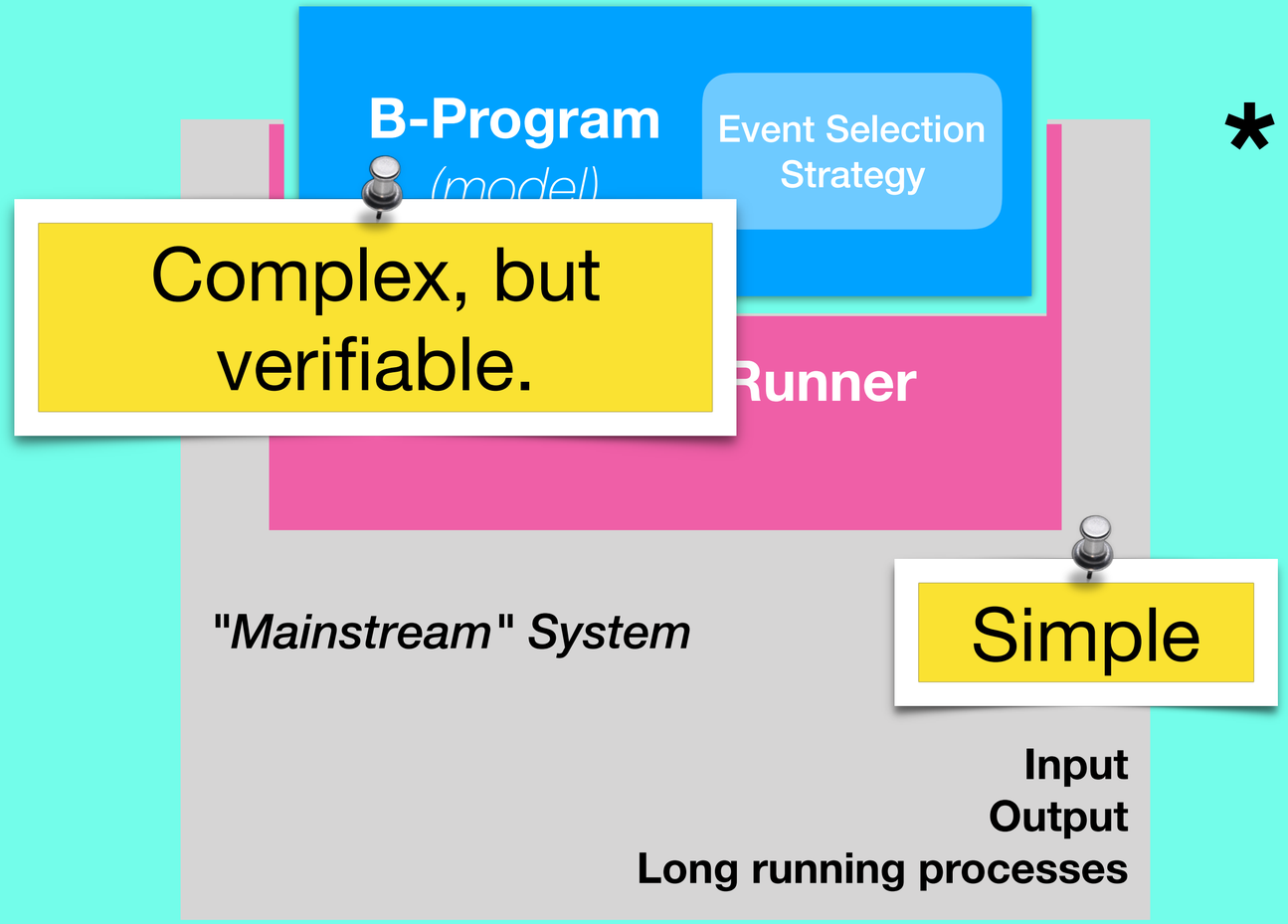
Why not Both?


B-Program
(model)
Event Selection Strategy
B-Program Runner
"Mainstream" System
Simple
Input
Output
Long running processes

Why not Both?


B-Program
(model)
Event Selection
Strategy
Complex, but
verifiable.
Runner
"Mainstream" System
Simple
Input
Output
Long running processes

Why not Both?



* Or, at least, partially verified, testable, and possibly runtime verified

Bruno Borges @brunoborges · 1d

Write the scariest tech presentation title you can using only 4 words.

I'll start:

356 131 274

Bruno Borges @brunoborges · 1d

Write the scariest tech presentation title you can using only 4 words.

I'll start:

356 131 274

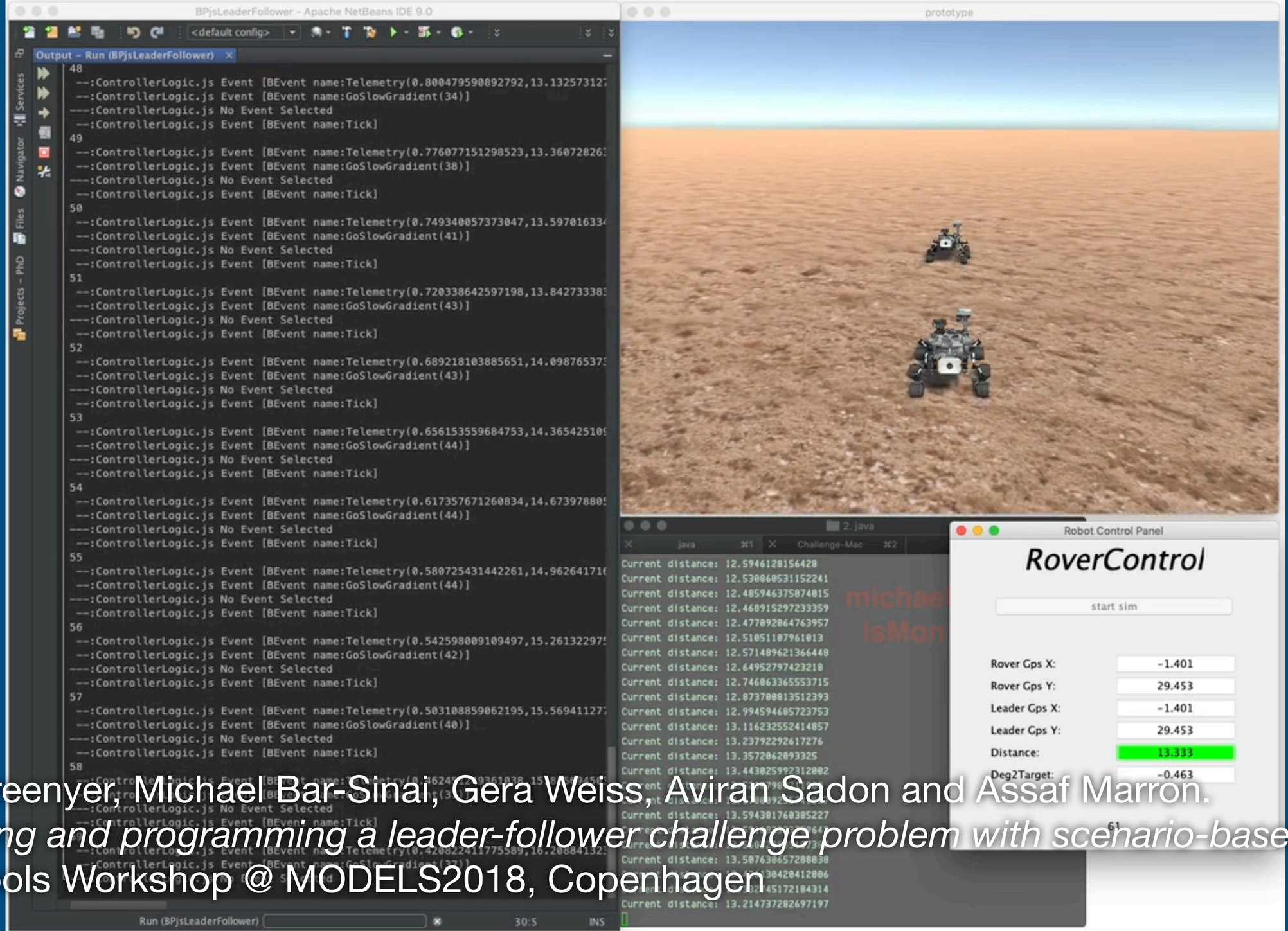
Simon Ritter
@speakjava

Replying to @brunoborges

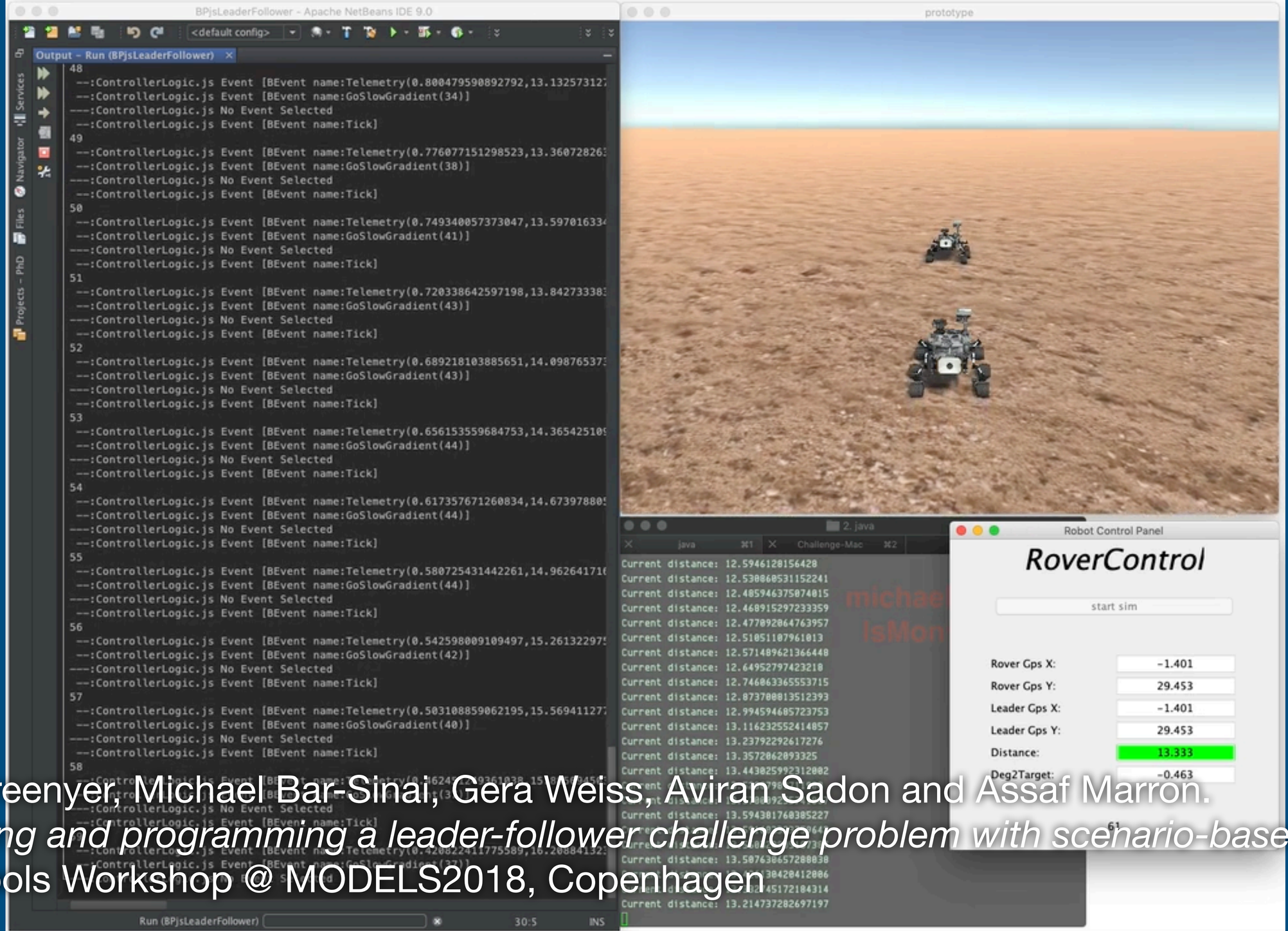
Javascript for autonomous vehicles.

11/10/2018, 21:31

14 Retweets **179** Likes

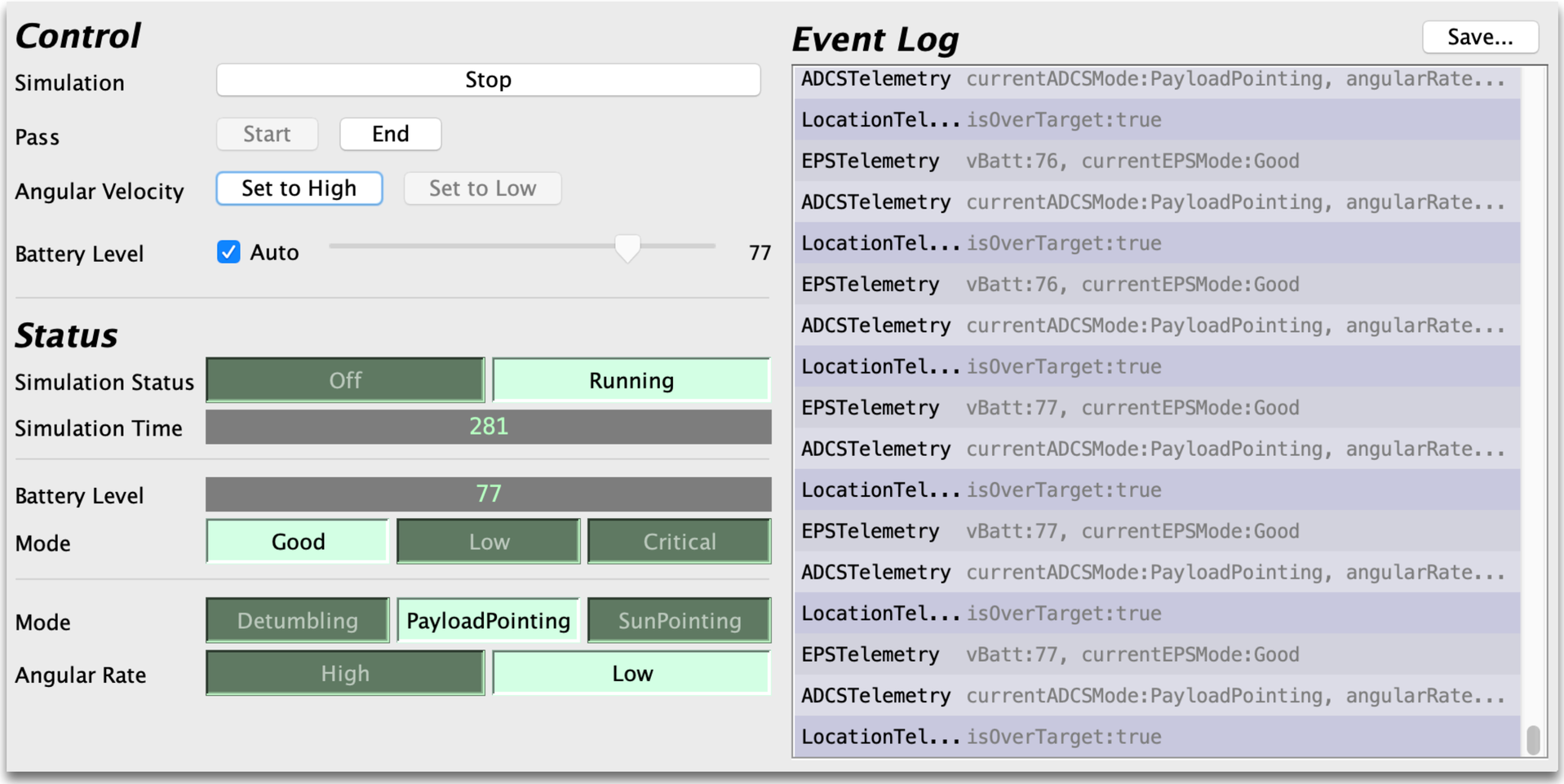


Joel Greenyer, Michael Bar-Sinai, Gera Weiss, Aviran Sadon and Assaf Marron.
Modeling and programming a leader-follower challenge problem with scenario-based tools.
MDETools Workshop @ MODELS2018, Copenhagen



Joel Greenyer, Michael Bar-Sinai, Gera Weiss, Aviran Sadon and Assaf Marron.
Modeling and programming a leader-follower challenge problem with scenario-based tools.
MDETools Workshop @ MODELS2018, Copenhagen

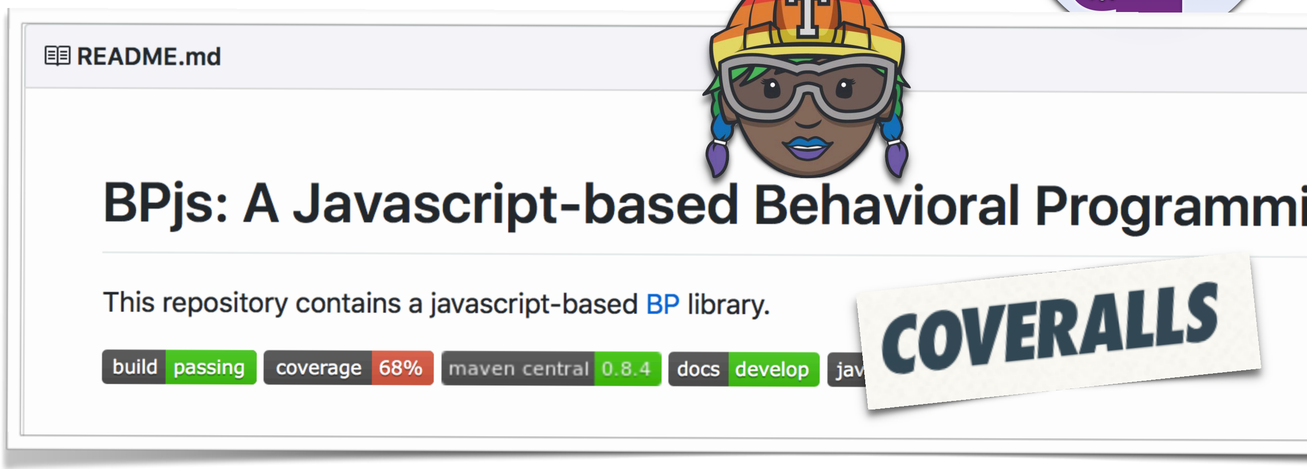
Space Applications



BPjs a BP Tool Suite for JS

- Open Source (MIT)
- Embeddable / Commandline
- Continuous UT+CC, Documentation
- GitHub Repo, Maven Central, Jars
- Sample code
- JS using Mozilla Rhino

Other tools out there too!




```
<dependency>
    <groupId>com.github.bthink-bgu</groupId>
    <artifactId>BPjs</artifactId>
    <version>0.9.3</version>
</dependency>
```

The Central Repository

Read the Docs
Create, host, and browse documentation.

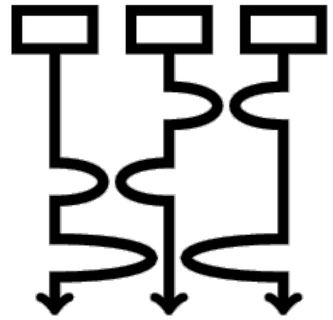
javadoc.io
104,770,580 page views since 2014-06-10 (updated daily).

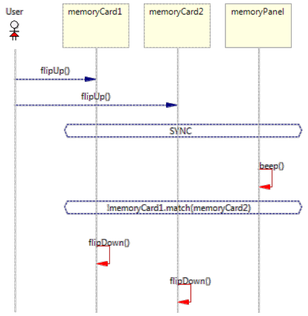
*mostly

People and Works

- **Scenario Based Programming**
  - Live Sequence Charts: **Werner Damm**, **David Harel** (1999, 2001)
  - **David Harel**, **Rami Marelly**, Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine (2003)
- **Behavioral Programming**
  - **David Harel**, **Assaf Marron**, **Gera Weiss** (2010, 2012)
- ***BPjs*** developed at BGU Software Engineering Group, **Gera Weiss** (2017)
  - **Michael Bar-Sinai**, **Aviran Sadon**, **Reut Shmuel**, **Moshe Weinstock**
  - Collaborating with **David Harel**'s Group, **Assaf Marron**
  - SE/CS Students
- Other BP Groups:
  - Weizmann Institute of Science (**David Harel**),
  - Leibniz Universität Hannover (**Joel Greenyer**)
  - JS/React: https://github.com/lmatteis/behavioral (**Luca Matteis**)

Resources





- **b-prog.org:** Main BP site, including other BP frameworks (C++, Java, Erlang, Blockly)
- **Academic Papers** (algorithms, event selections, visualizations, ...) *Behavioral Programming, Live Sequence Charts (LSCs)*
- **github.com/bthink-BGU:** BPjs, projects presented in this talk
- **bpjs.readthedocs.io:** BPjs tutorial and reference
- **groups.google.com/forum/#!forum/bpjs:** Google group
- **scenariotools.org** Live Sequence Charts take on this subject, Joel Greenyer

ScenarioTools.org

DEVOXX

Please Rate Me!

If you enjoyed the talk
Or give feedback

Ben-Gurion University of the Negev

```
// Thanks.
bp.sync({
  waitFor:questions
});
```

@michbarsinai
http://b-prog.org
http://github.com/bthink-BGU

Partial funding for this presentation generously granted by the Frenkel Foundation, BGU