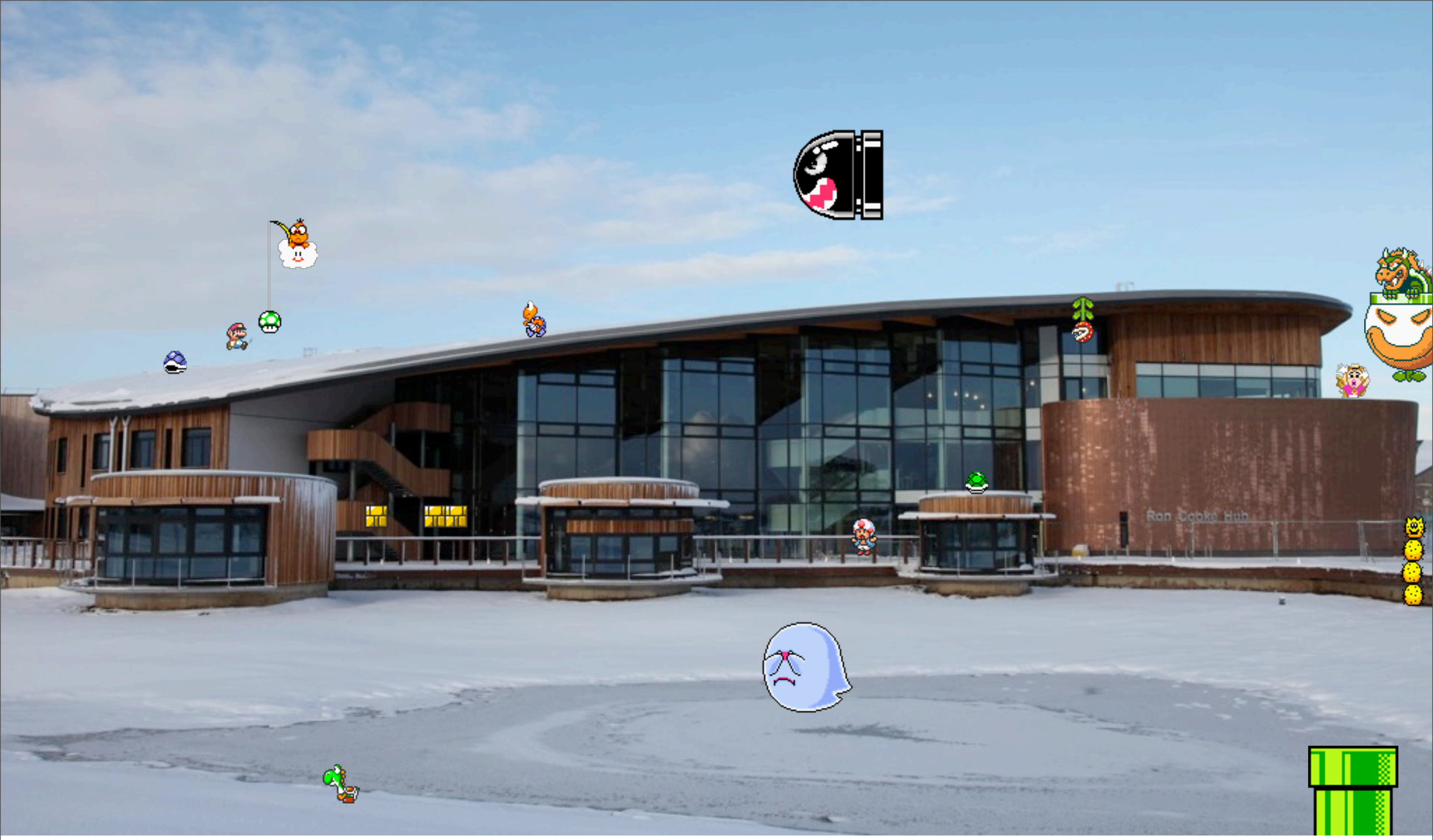


Identifying Desirable Game Character Behaviours through the Application of Evolutionary Algorithms to Model-Driven Engineering Metamodels

James R. Williams, Simon Poulding,
Louis M, Rose, Fiona A. C. Polack,
Richard F. Paige

University of York, UK

Novel application of GE in combination with MDE

MDE and DSLs

Genotype-to-Phenotype: A Model Transformation

Case Study: SAF

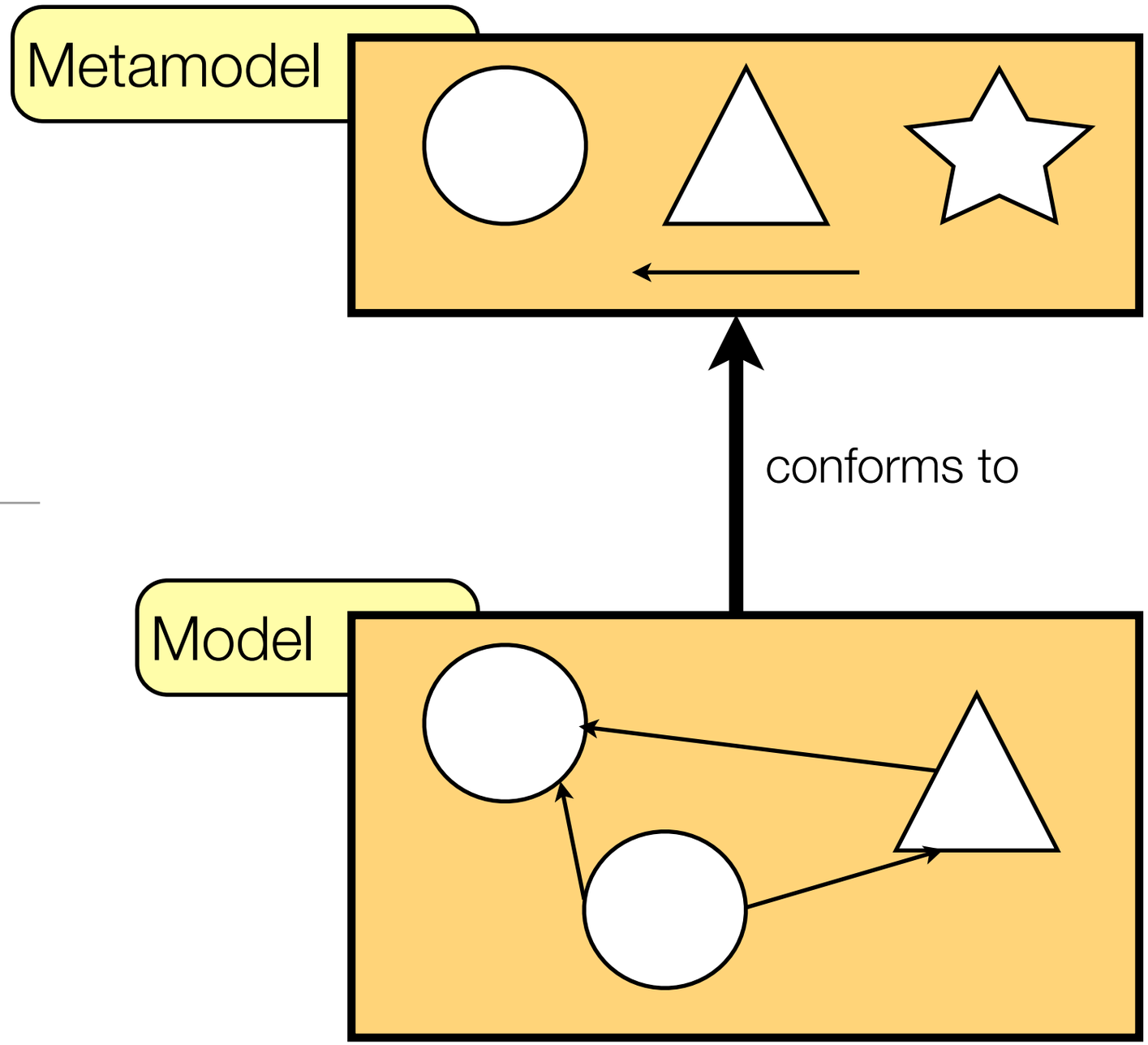
MDE and DSLs

Genotype-to-Phenotype: A Model Transformation

Case Study: SAF

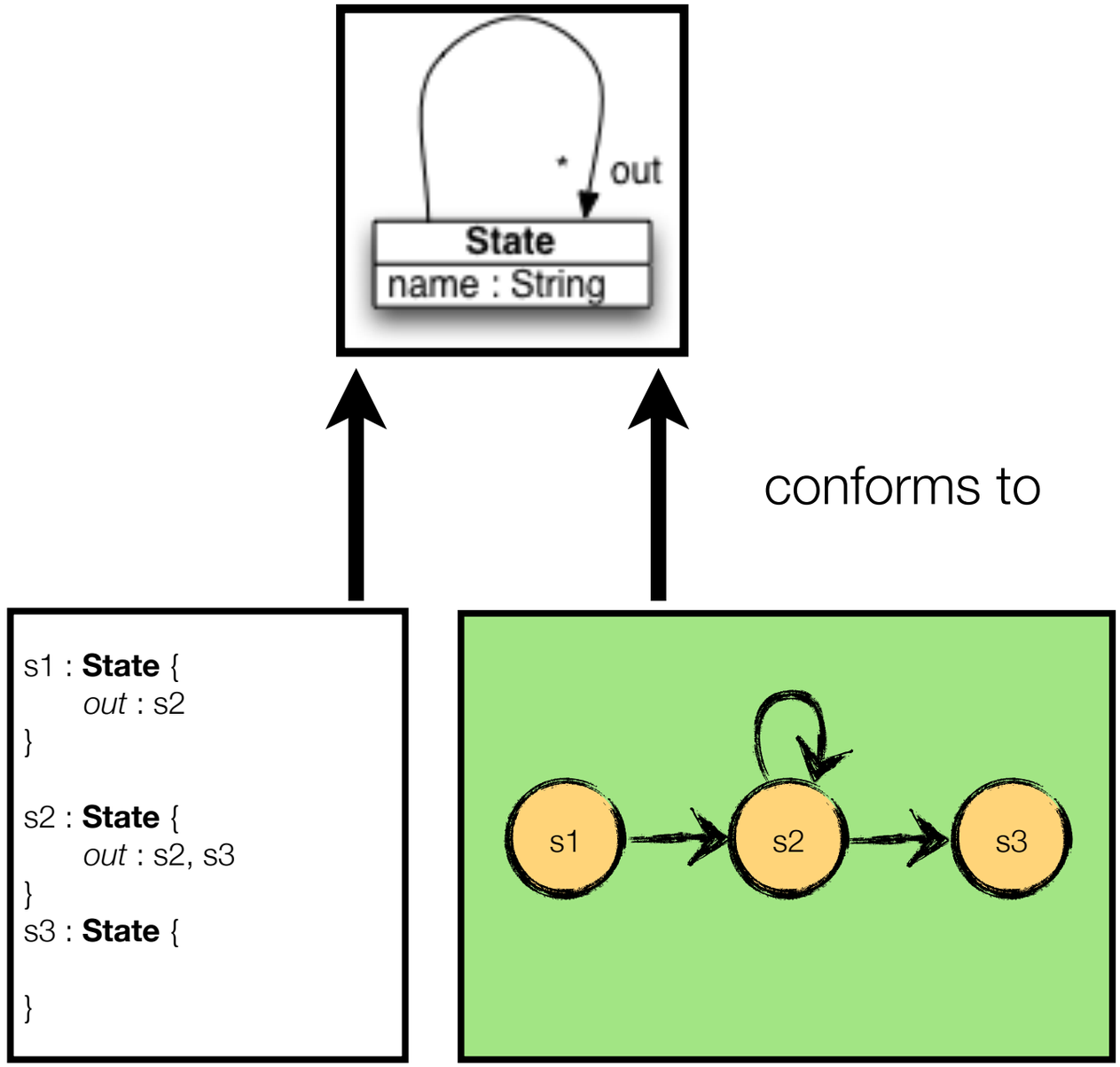
Model Driven Engineering

Treats models as first-class artefacts in the software development lifecycle.



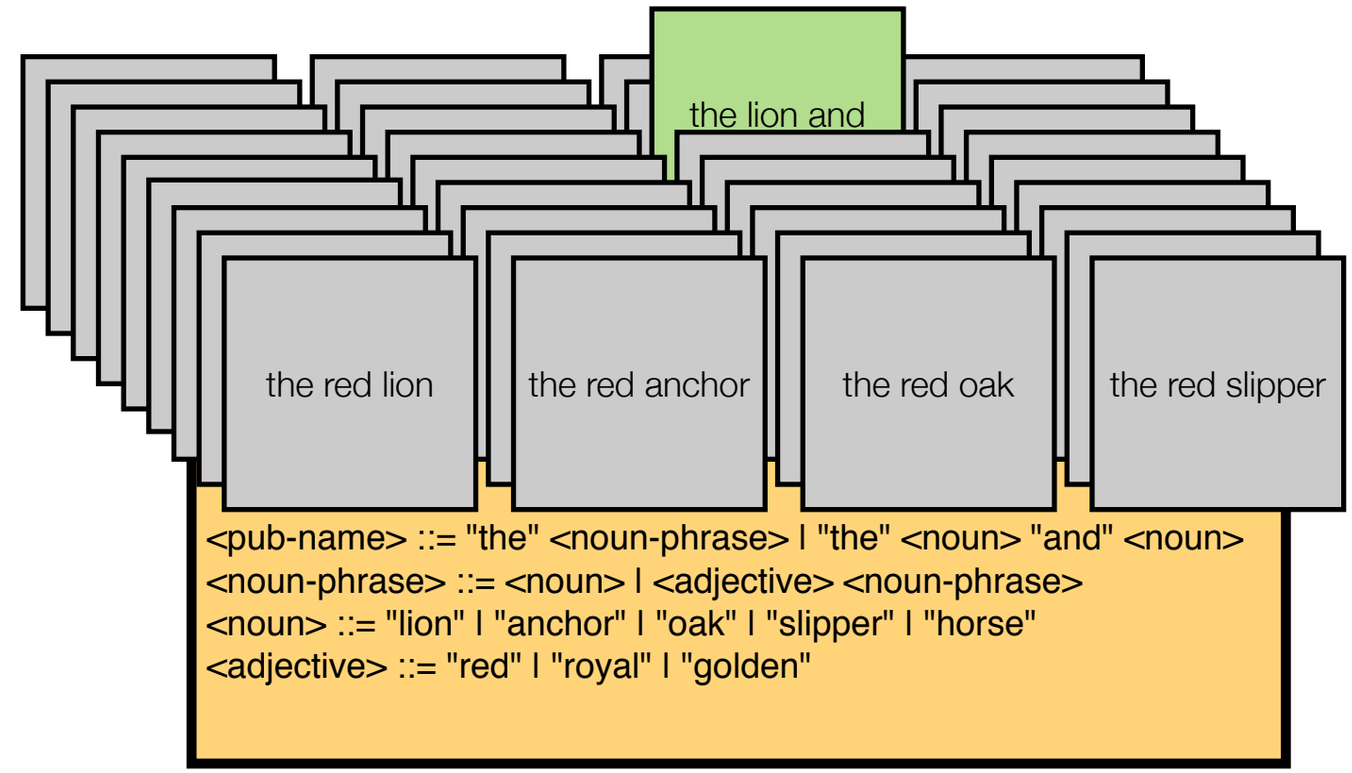
Domain-Specific Languages

Models can be graphical, textual, or both.



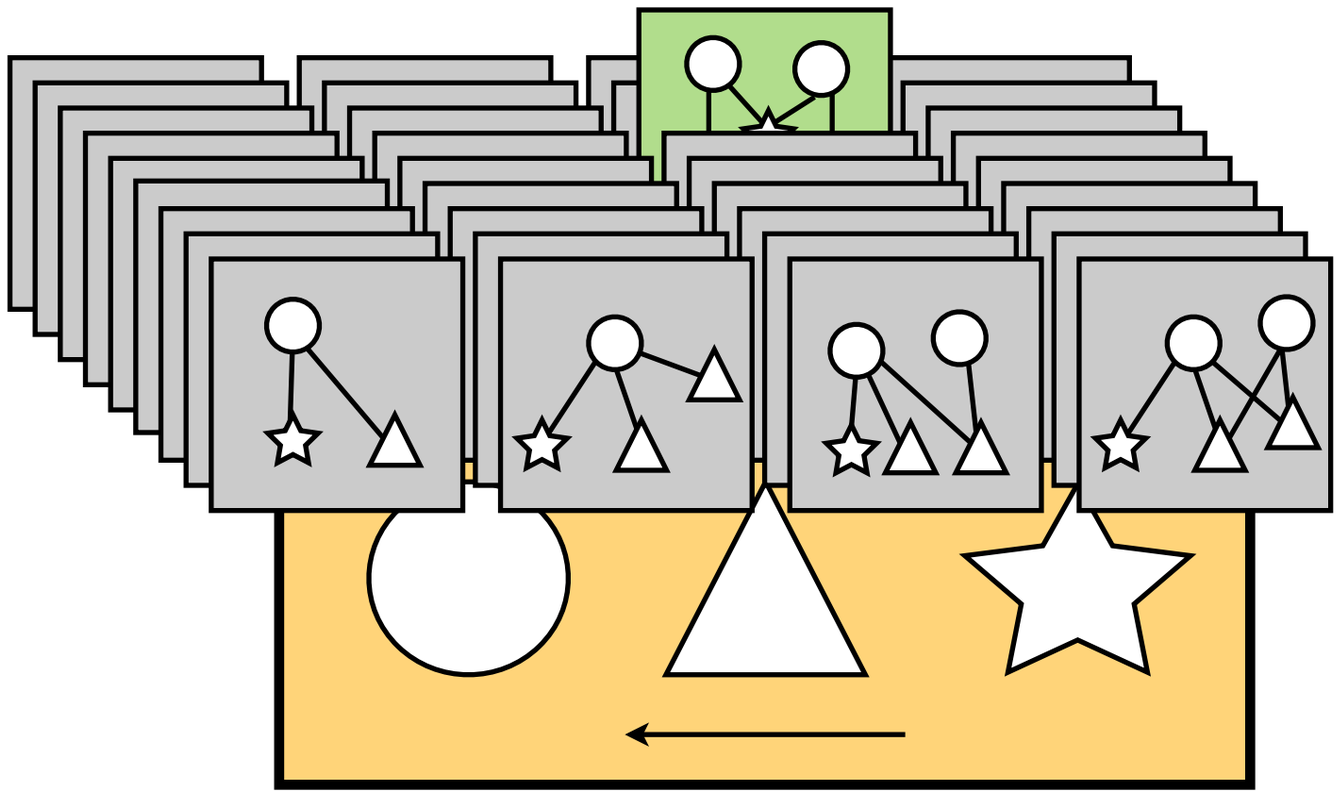
Search

Grammars define a space of valid strings over which we can search.



Search

Metamodels define a space of valid models over which we can search.



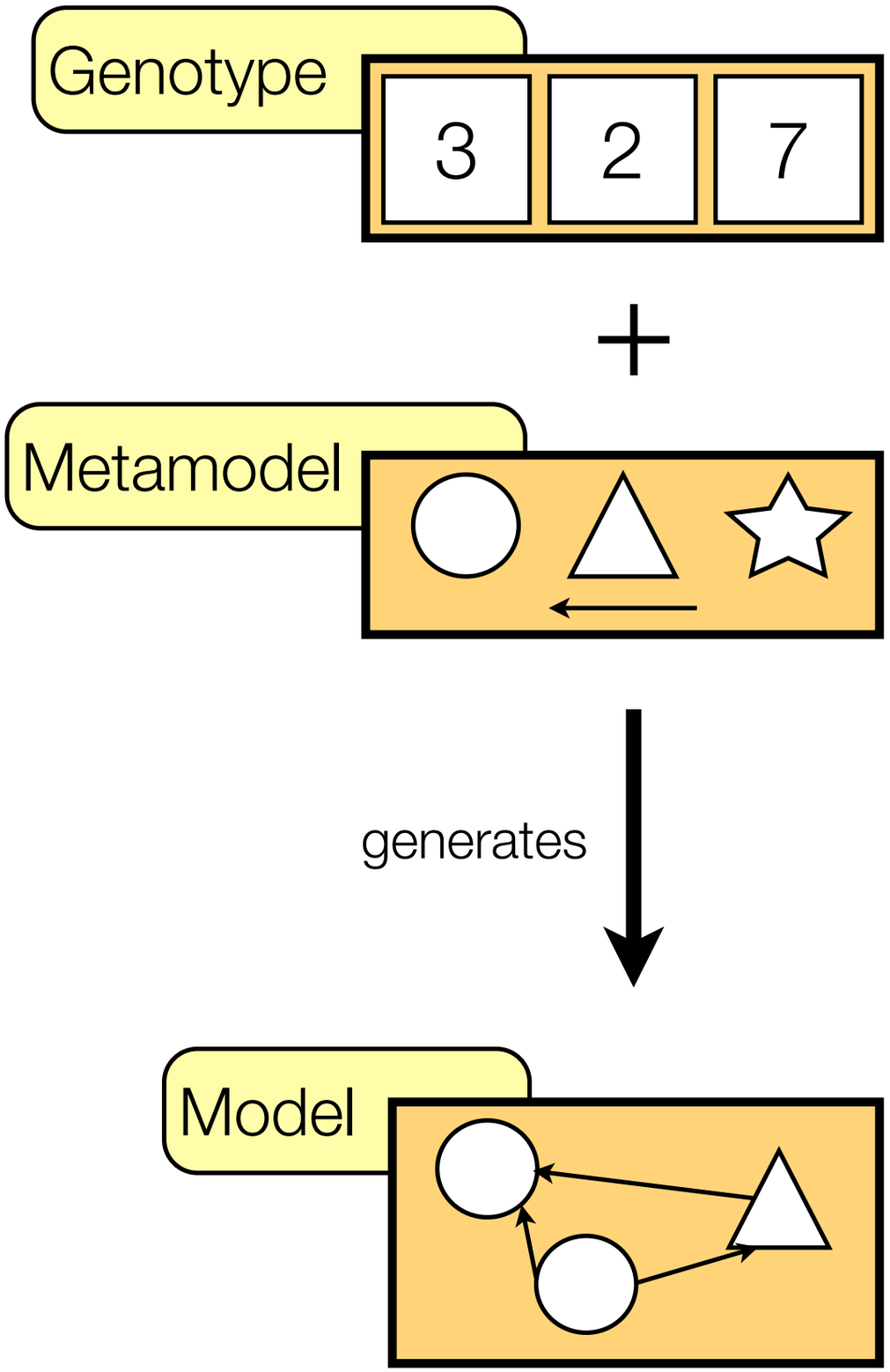
MDE and DSLs

Genotype-to-Phenotype: A Model Transformation

Case Study: SAF

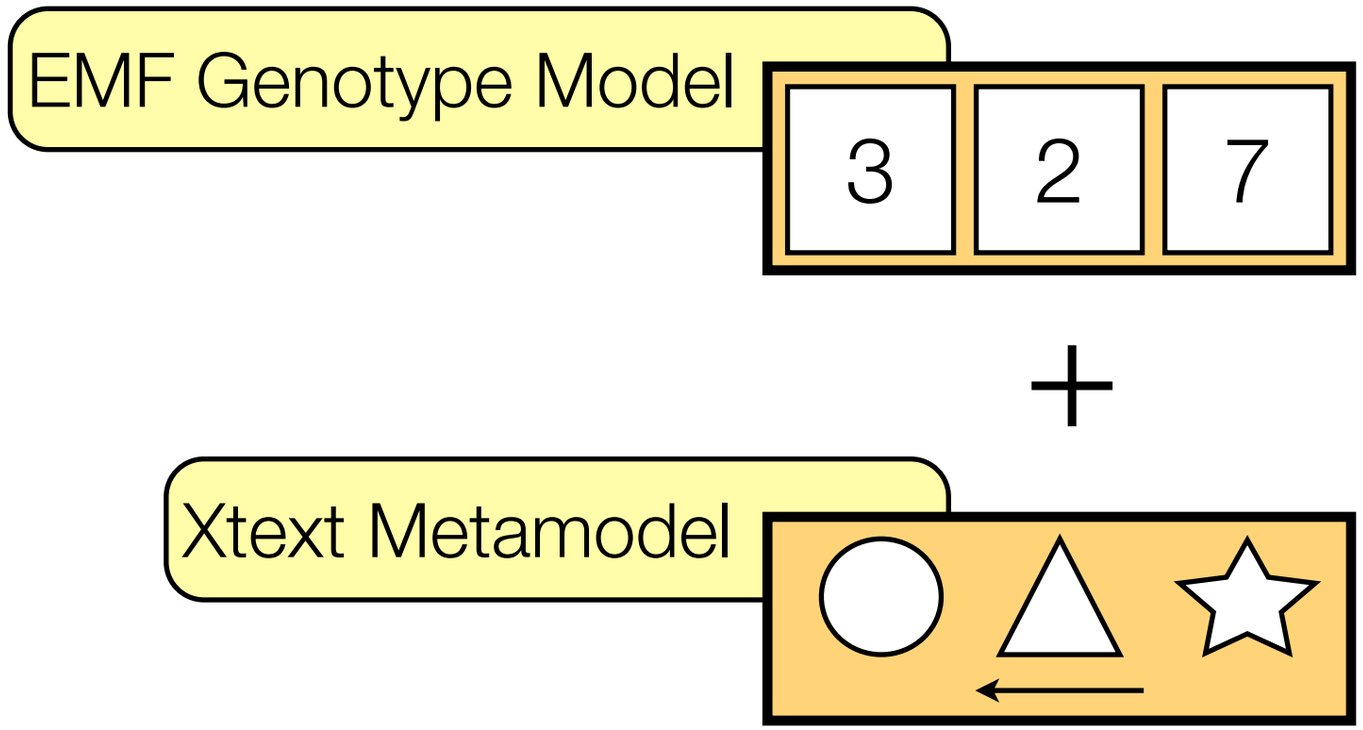
Technique

Grammatical evolution-like mapping from genotype to phenotype.

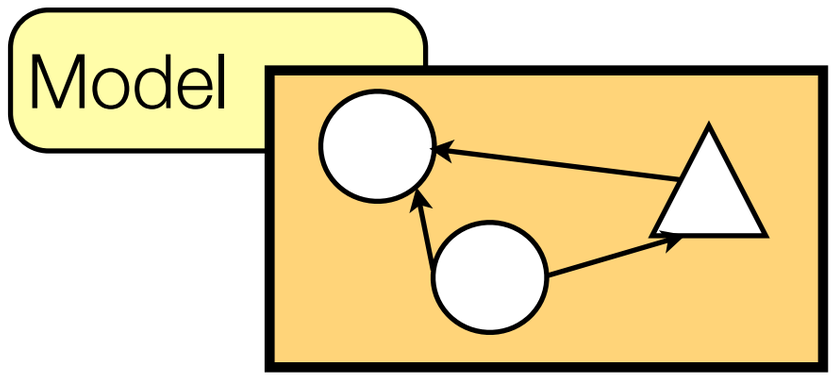


Implementation

Eclipse Modeling Framework
Xtext
Epsilon



model to text transformation
using Epsilon



www.eclipse.org/xtext
www.eclipse.org/emf
www.eclipse.org/gmt/epsilon/

MDE and DSLs

Genotype-to-Phenotype: A Model Transformation

Case Study: SAF

Human player FDL

```
fighter human {
  punchPower=7
  kickReach=4
  near[crouch punch_low]
  always[walk_away kick_high]
}
```

Super Awesome Fighter 4000

versus

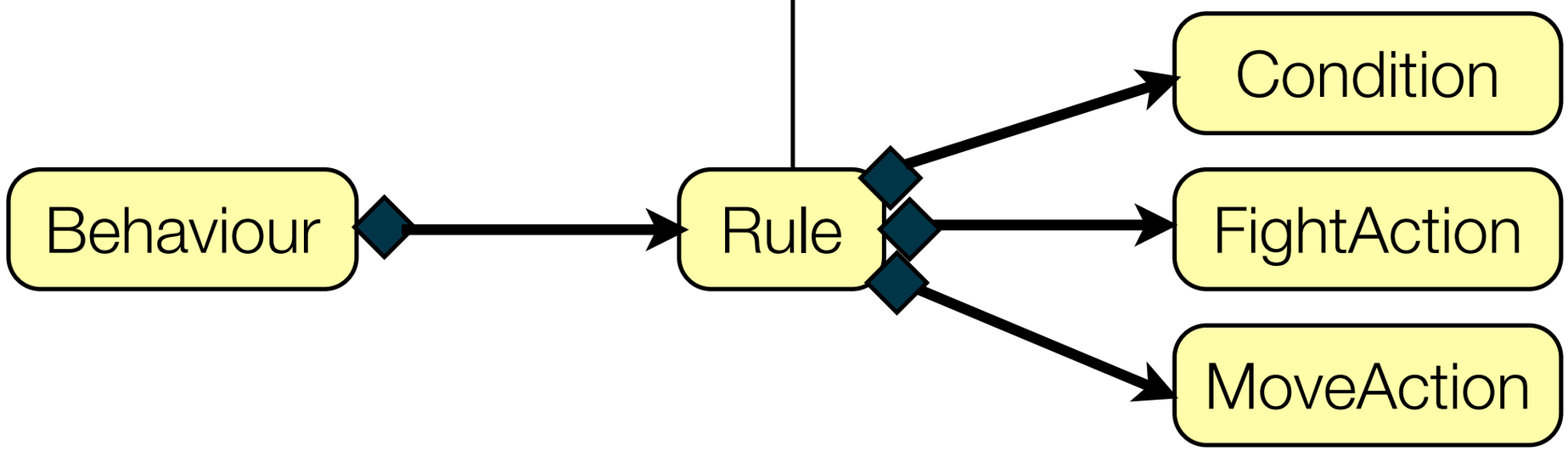
Non-player FDL

```
fighter nonPlayer {
  punchReach=8
  near[jump kick_high]
  far[run_towards block_low]
  always[run_away block_high]
}
```

Fighter Description Language

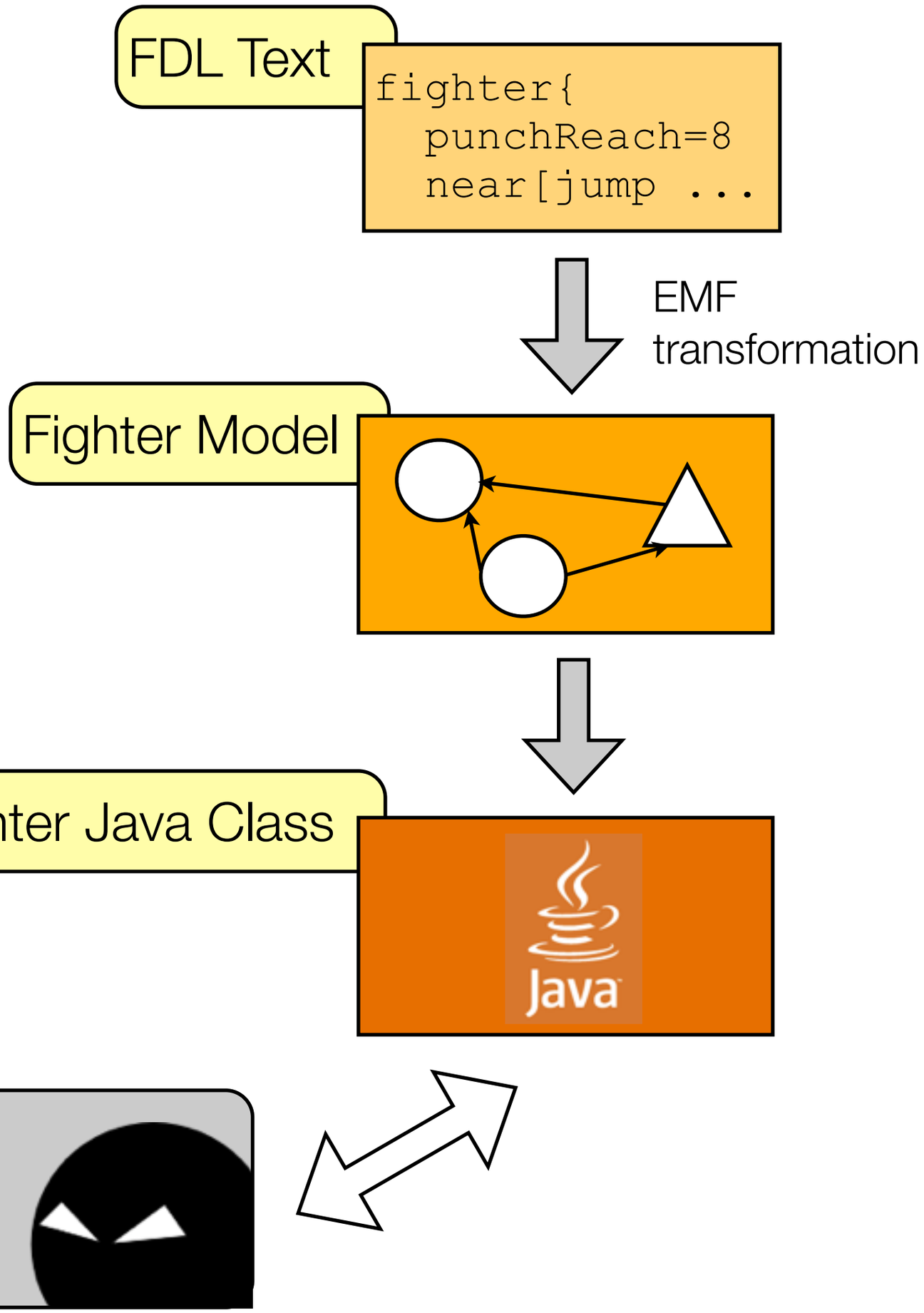
A language for specifying the behaviour of characters in SAF.


```
fighter {
  kickReach=8
  kickPower=7
  punchPower=3
  punchReach=9
  far or much_weaker[crouch punch_high]
  near[jump choose(kick_low block_low)]
  always[crouch kick_high]
}
```



Fighter Description Language

A language for specifying the behaviour of characters in SAF.

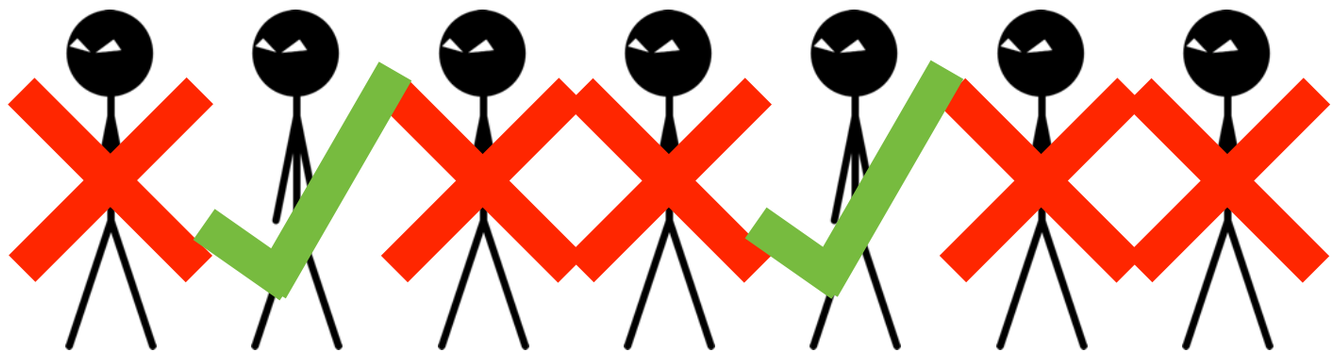


Engineering Objective

Searching for non-player fighters with desirable properties.

Non-player Fighter

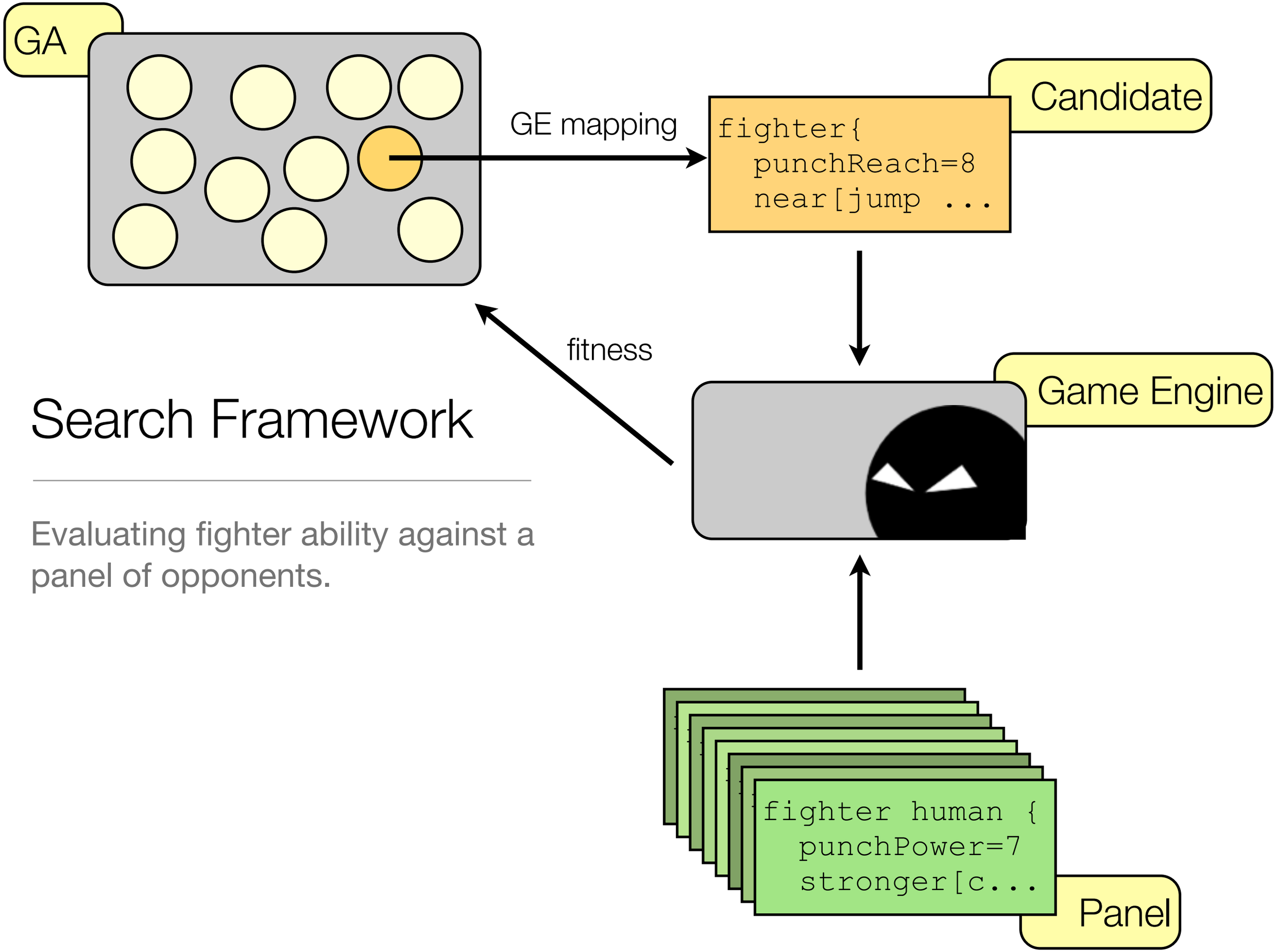
```
fighter{
  punchReach=8
  near[jump ...
```



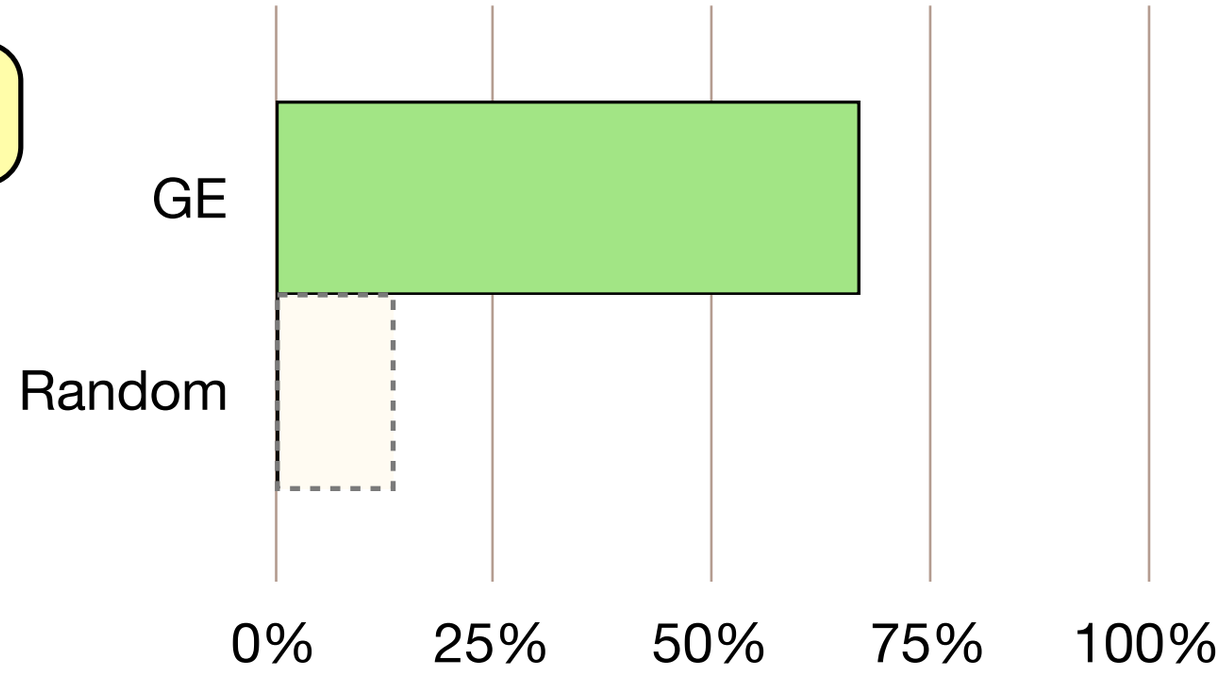
Human Opponents

Search Framework

Evaluating fighter ability against a panel of opponents.



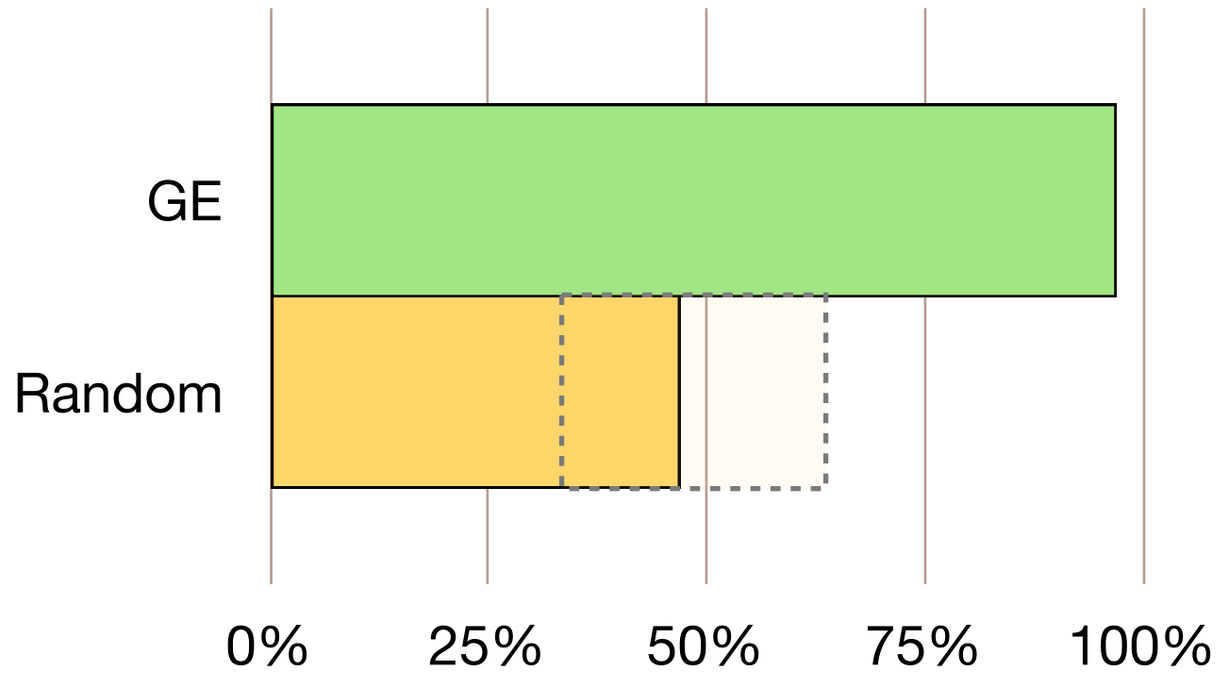
Non-Player Wins 100%



Results

Does search method facilitate the engineering objectives?

Non-Player Wins 80%



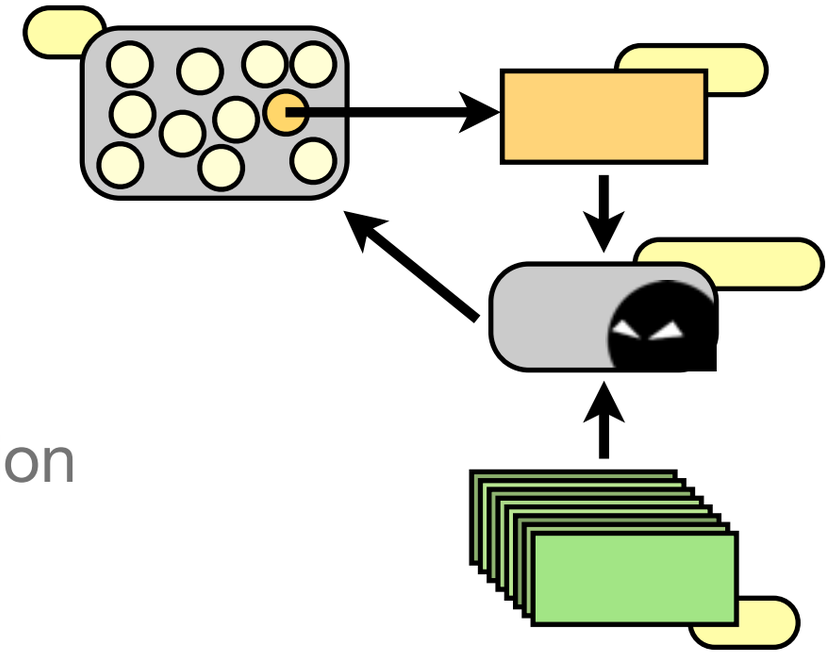
Results

Some unexpected – and very useful – outcomes.

```
fighter{
   punchPower=9
   punchPower=7
   punchPower=2
   kickPower=7
   punchPower=2
   kickPower=2
   near[crouch punch_low]
   stronger or far[choose(run_towards
      run_towards) kick_high]
   much_weaker and weaker[walk_away
      block_low]
   always[crouch kick_high]
}
```

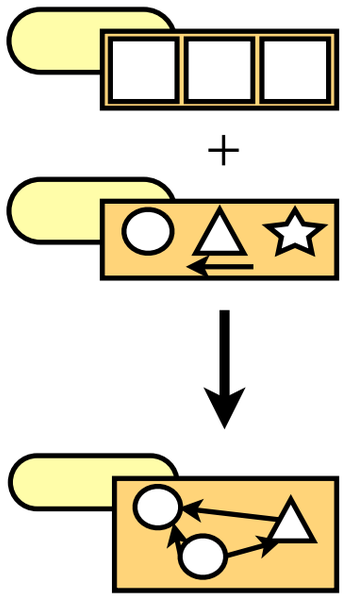
Conclusion and Future Work

- Novel application of GE



- Genotype-to-phenotype model transformation

  - Generic to any metamodel



Future Work

- Improvements to the genotype-phenotype transformation
- Opponents derived using co-evolutionary methods
- Dynamic evolution
- Bidirectional transformation

MDE and DSLs

Genotype-to-Phenotype: A Model Transformation

Case Study: SAF

Thank you!