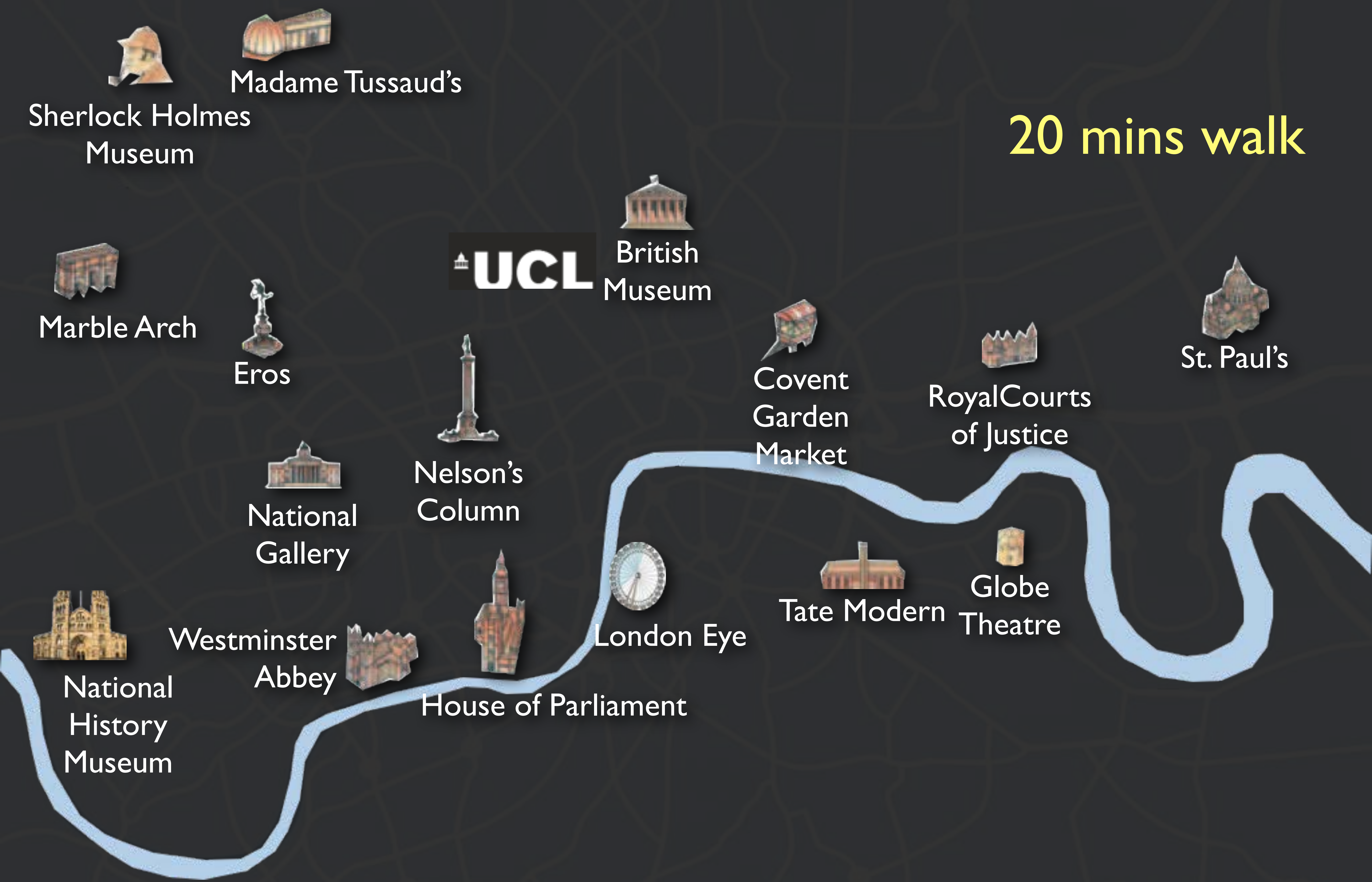
SEAMS2014

Genetic Improvement for Adaptive Software Engineering

Mark Harman

joint work with Yue Jia, Bill Langdon, Iman Moghadam,
Justyna Petke, Shin Yoo & Fan Wu

University College London



Madame Tussaud's
Sherlock Holmes Museum
20 mins walk
UCL
British Museum
Marble Arch
Eros
Covent Garden Market
RoyalCourts of Justice
St. Paul's
National Gallery
Nelson's Column
London Eye
Tate Modern
Globe Theatre
Westminster Abbey
National History Museum
House of Parliament

COWs

CREST Open Workshop

Roughly one per month

Discussion based

Recorded and archived

COWs

CREST Open Workshop

Roughly one per month

Discussion based

Recorded and archived

COWs

CREST Open Workshop

Roughly one per month

Discussion based

Recorded and archived

COWs


Stockholm
Tallinn
Baltic Sea
Estonia
Latvia
Lithuania
Moscow
North Sea
Denmark
Copenhagen
Vilnius
Minsk
Belarus
Ireland
Berlin
Poland
Germany
Prague
Kiev
Kharkiv
Czech Rep
Kraków
Slovakia
Ukraine
Paris
Dnipropetrovs'k
Moldova
France
Odesa
Bay of Biscay
Croatia
Turin
Serbia
Bucharest
Black Sea
Marseille
Rome
Bulgaria
Georgia
Barcelona
Portugal
Spain
Naples
Ankara
Turkey
Izmir
Athens
Algiers
Tunis
Nova Scotia

What is SBSE

What is SBSE

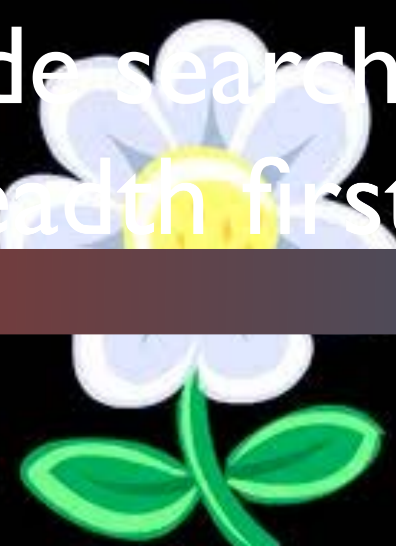
In SBSE we apply search techniques to search large search spaces, guided by a fitness function that captures properties of the acceptable software artefacts we seek.

sweet spot

like google search?

like code search?

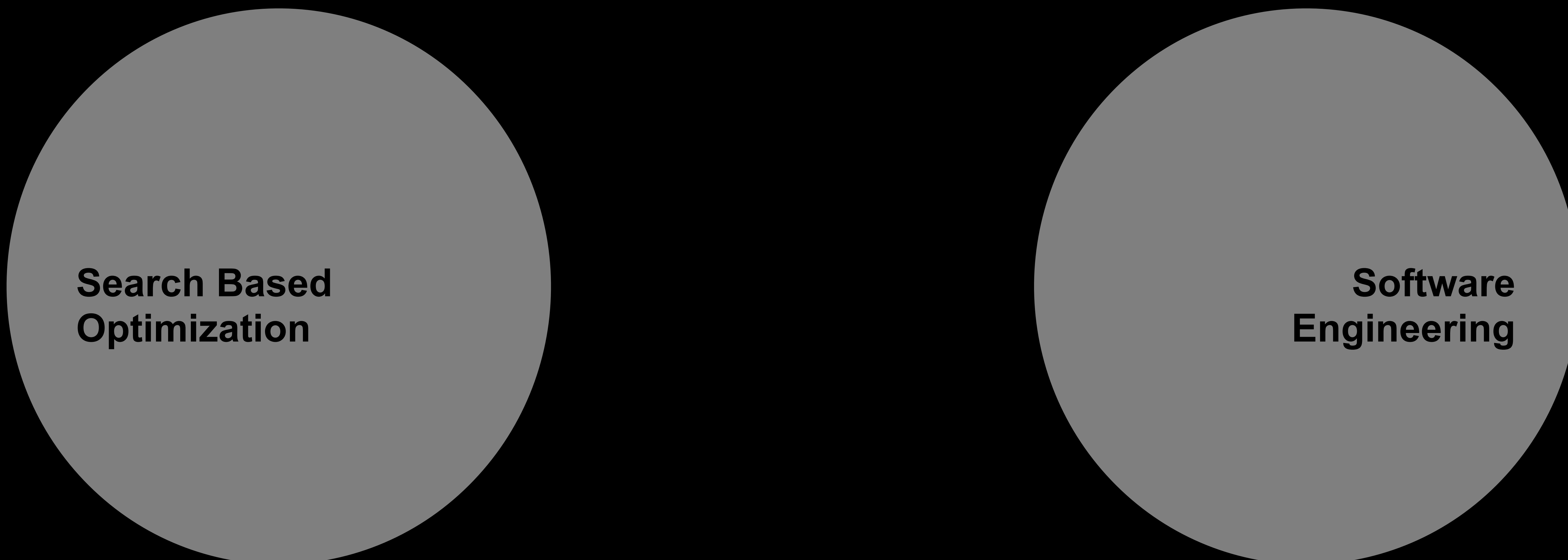
like breadth first search?



potentially exhaustive

pick one at random

What is SBSE



What is SBSE

In SBSE we apply search techniques to search large search spaces, guided by a fitness function that captures properties of the acceptable software artefacts we seek.

Tabu Search
Ant Colonies
Particle Swarm Optimization
Genetic Algorithms
Hill Climbing
Genetic Programming
Simulated Annealing
Greedy
LP
Random
Estimation of Distribution Algorithms

What is SBSE

In SBSE we apply search techniques to search large search spaces, guided by a fitness function that captures properties of the acceptable software artefacts we seek.

Tabu Search
Ant Colonies
Particle Swarm Optimization
Genetic Algorithms
Hill Climbing
Genetic Programming
Simulated Annealing
Random
Greedy
LP
Estimation of Distribution Algorithms

What is SBSE

let's listen to software engineers ...

... what sort of things do they say?

Software Engineers Say

We need to satisfy business and technical concerns

We need to reduce risk while maintaining completion time

We need increased cohesion and decreased coupling

We need fewer tests that find more nasty bugs

We need to optimise for all metrics M1,..., Mn

Software Engineers Say

We need to satisfy business and technical concerns

We need to reduce risk while maintaining completion time

We need increased cohesion and decreased coupling

We need fewer tests that find more nasty bugs

We need to optimise for all metrics M1,...., Mn

Software Engineers Say

We need to satisfy business and technical concerns

We need to reduce risk while maintaining completion time

We need increased cohesion and decreased coupling

We need fewer tests that find more nasty bugs

We need to optimise for all metrics M1,...., Mn

Software Engineers Say

We need to satisfy business and technical concerns

We need to reduce risk while maintaining completion time

We need increased cohesion and decreased coupling

We need fewer tests that find more nasty bugs

We need to optimise for all metrics M1,...., Mn

Software Engineers Say

We need to satisfy business and technical concerns

We need to reduce risk while maintaining completion time

We need increased cohesion and decreased coupling

We need fewer tests that find more nasty bugs

We need to optimise for all metrics M1,...., Mn

Software Engineers Say

We need to satisfy business and technical concerns

We need to reduce risk while maintaining completion time

We need increased cohesion and decreased coupling

We need fewer tests that find more nasty bugs

We need to optimise for all metrics M1,..., Mn

Software Engineers Say

Requirements: We need to satisfy business and technical concerns

Management: We need to reduce risk while maintaining completion time

Design: We need increased cohesion and decreased coupling

Testing: We need fewer tests that find more nasty bugs

Refactoring: We need to optimise for all metrics M1,..., Mn

Software Engineers Say

Requirements: We need to satisfy business and technical concerns

Management: We need to reduce risk while maintaining completion time

Design: We need increased cohesion and decreased coupling

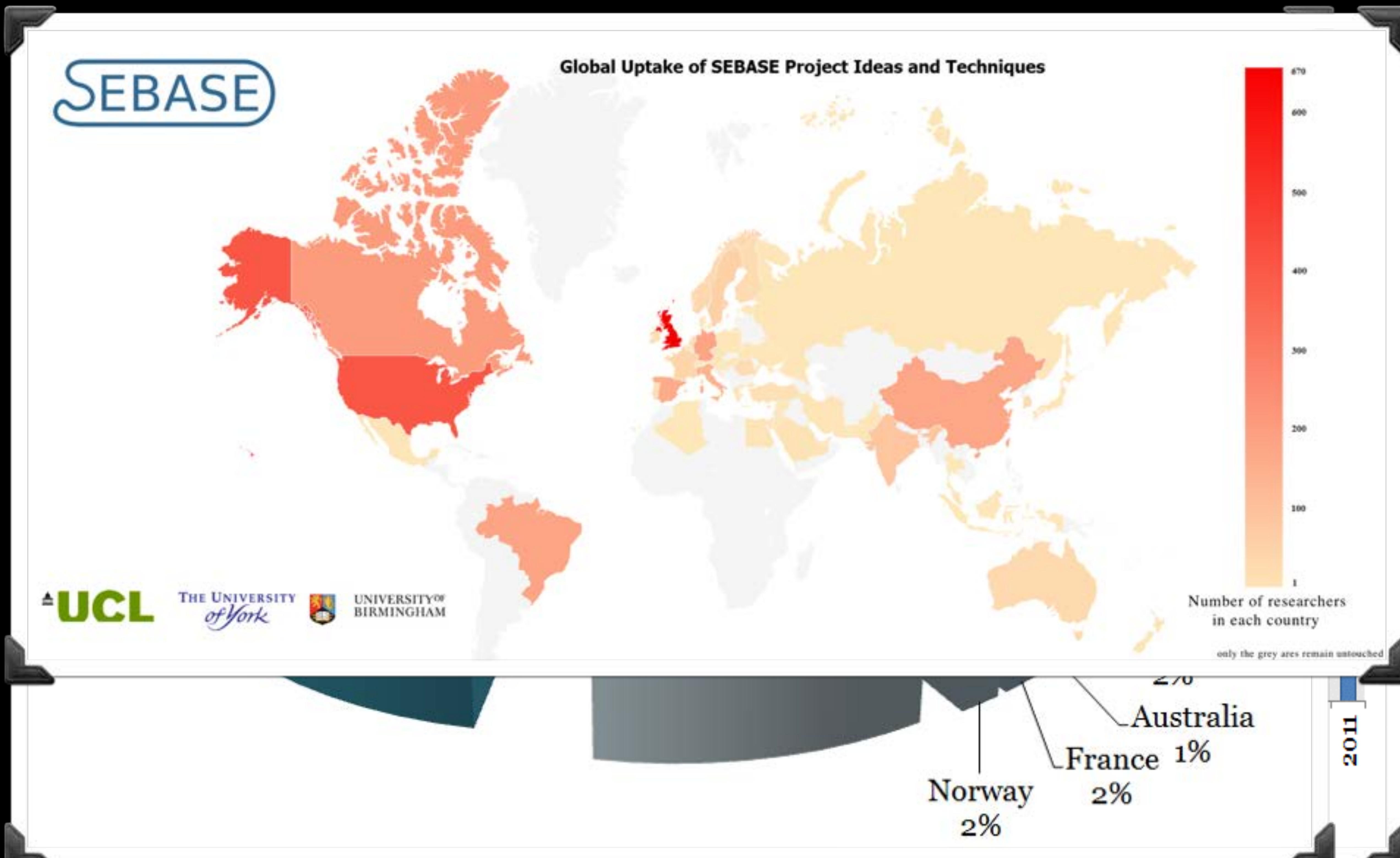
Testing: We need fewer tests that find more nasty bugs

Refactoring: We need to optimise for all metrics M1,..., Mn

All have been addressed in the SBSE literature

Growth Trends

Polynomial rise in publications




Dalian Liaoning China
CSBSE2013
2nd
Chinese Search Based Software Engineering Workshop | Dalian, China | 2013
WESB 2012
3º Workshop de Engenharia de Software Baseada em Buscas
23 de Setembro de 2012 | Natal-RN-Brasil
GECCO
Genetic and Evolutionary Computation Conference
2011
ssbse
4th Symposium on
Search Based Software Engineering
September 28th - 30th, 2012
Riva del Garda | Trento | Italy
INFORMATION AND SOFTWARE TECHNOLOGY
computers & operations research
JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION RESEARCH AND PRACTICE
EMPIRICAL SOFTWARE ENGINEERING
An International Journal
Springer
SOFTWARE PRACTICE & EXPERIENCE
IEEE TRANSACTIONS ON SOFTWARE ENGINEERING
The Journal of Systems and Software
4th International Workshop on
Search-Based Software Testing
March, 2011, Berlin, Germany
In conjunction with ICST 2011
IEEE International Conference on Testing, Verification and Validation
cmsbse
2013

Author statistics

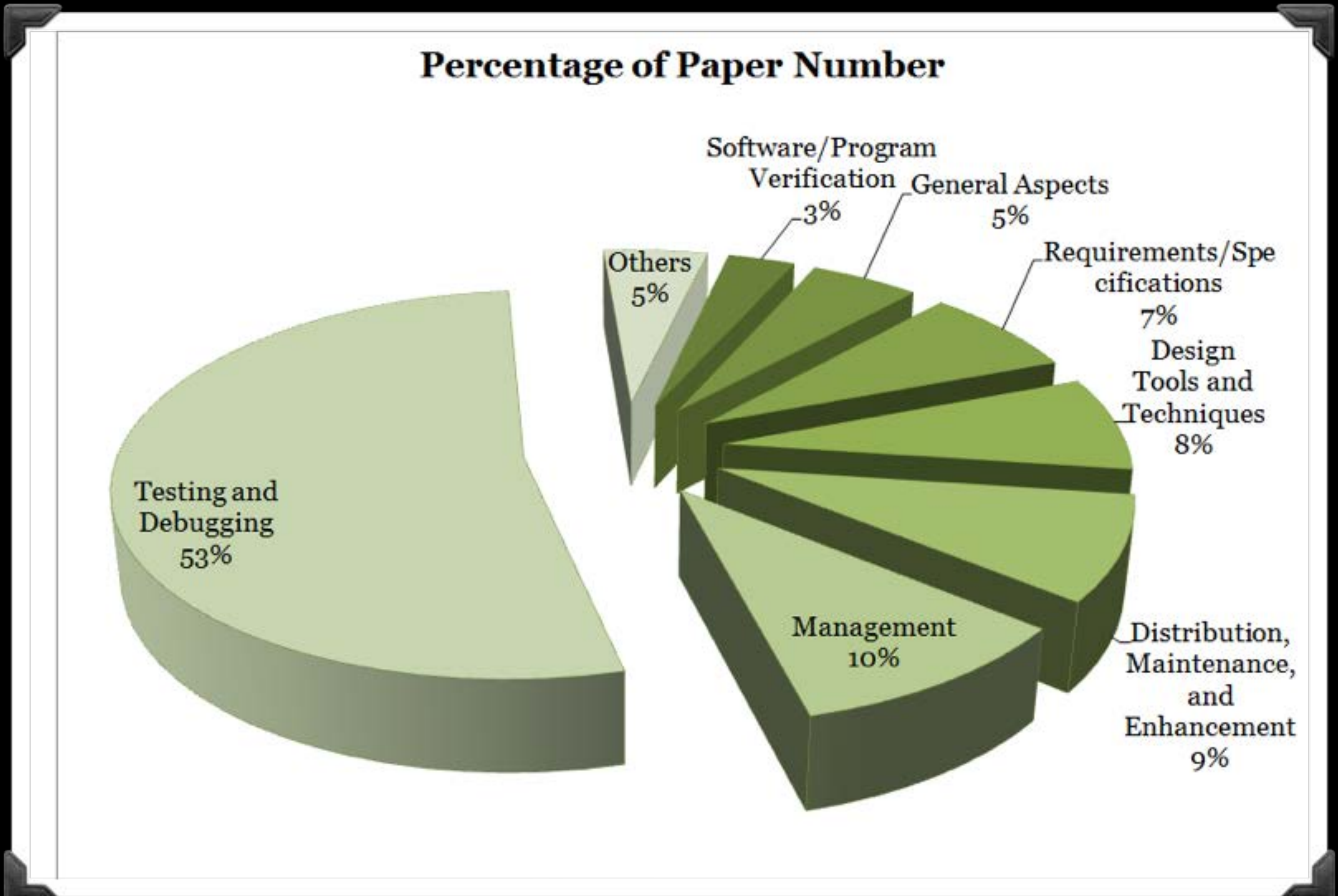
more than 1250 authors

more than 1150 papers

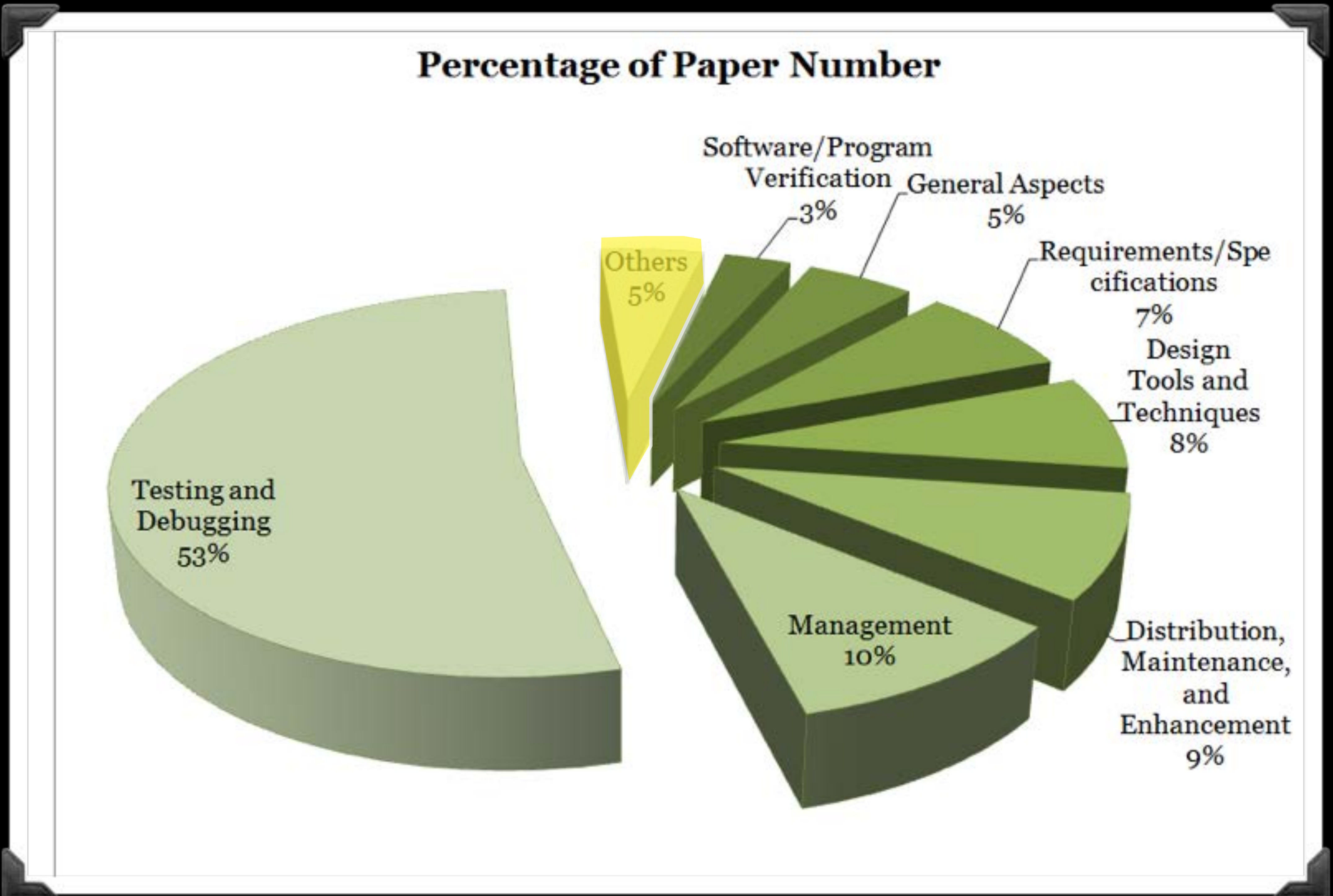
more than 390 institutions

more than 50 countries

source: SBSE repository, July 2013.



Software Engineering topics attacked



Software Engineering topics attacked

Just some of the many SBSE applications

Agent Oriented
Aspect Oriented
Assertion Generation
Bug Fixing
Component Oriented
Design
Effort Estimation
Heap Optimisation
Model Checking
Predictive Modelling
Probe distribution
Program Analysis
Program Comprehension
Program Transformation
Project Management
Protocol Optimisation
QoS
Refactoring
Regression Testing
Requirements
Reverse Engineering
SOA
Software Maintenance and Evolution
Test Generation
UIO generation

EPSRC
Grant

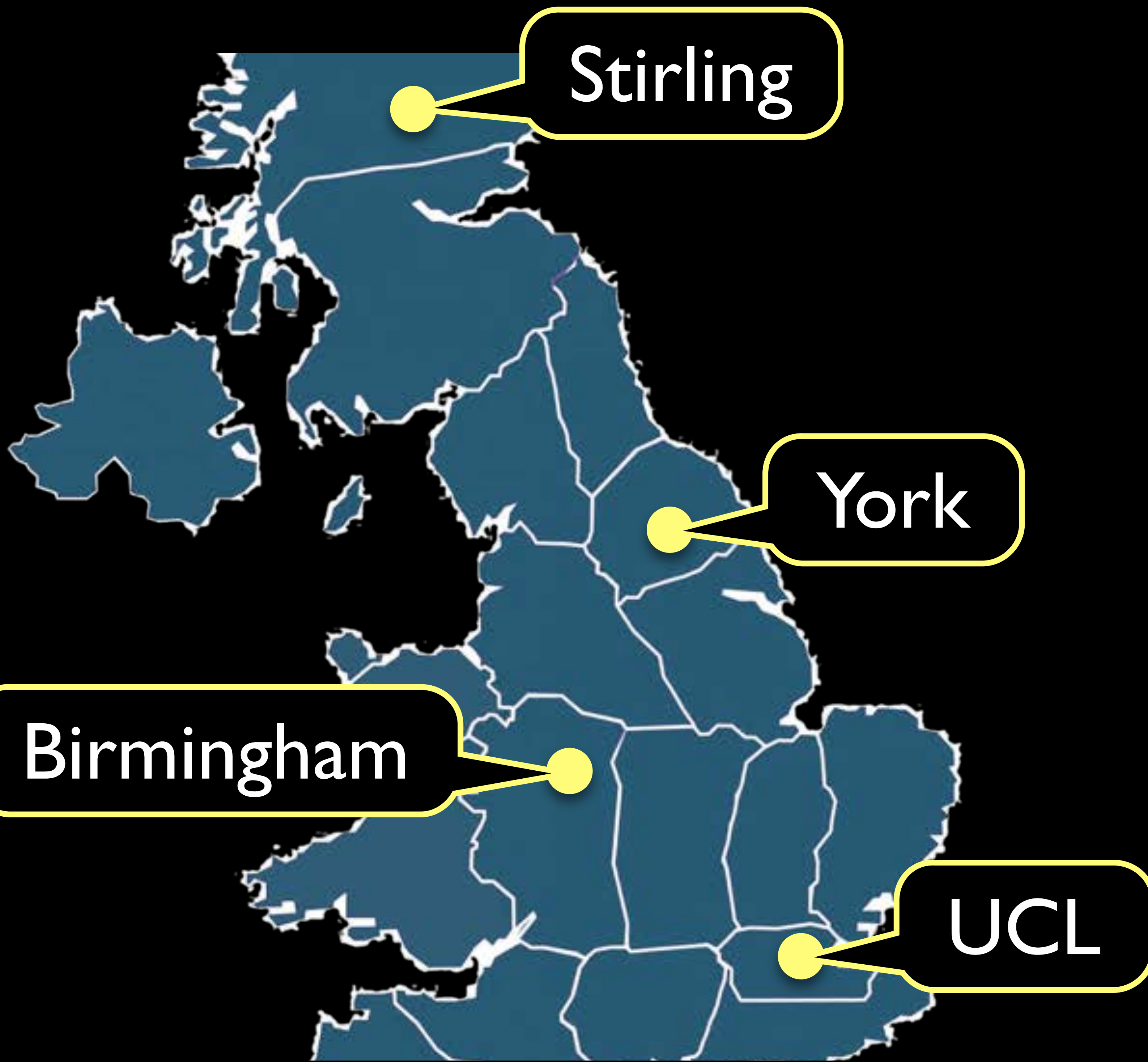
UCL

UNIVERSITY OF STIRLING

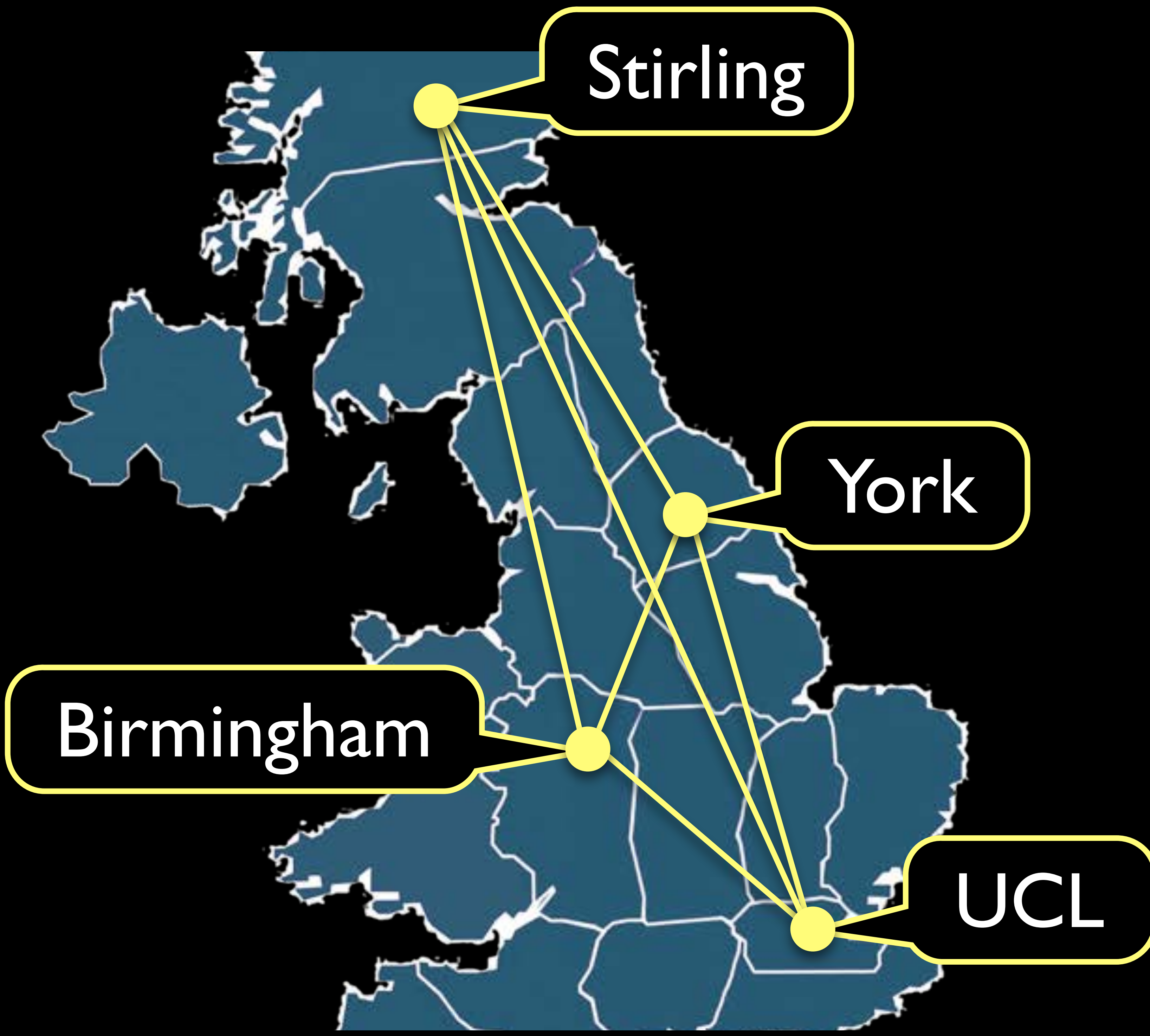
UNIVERSITY OF BIRMINGHAM

THE UNIVERSITY of York

EPSRC
Grant



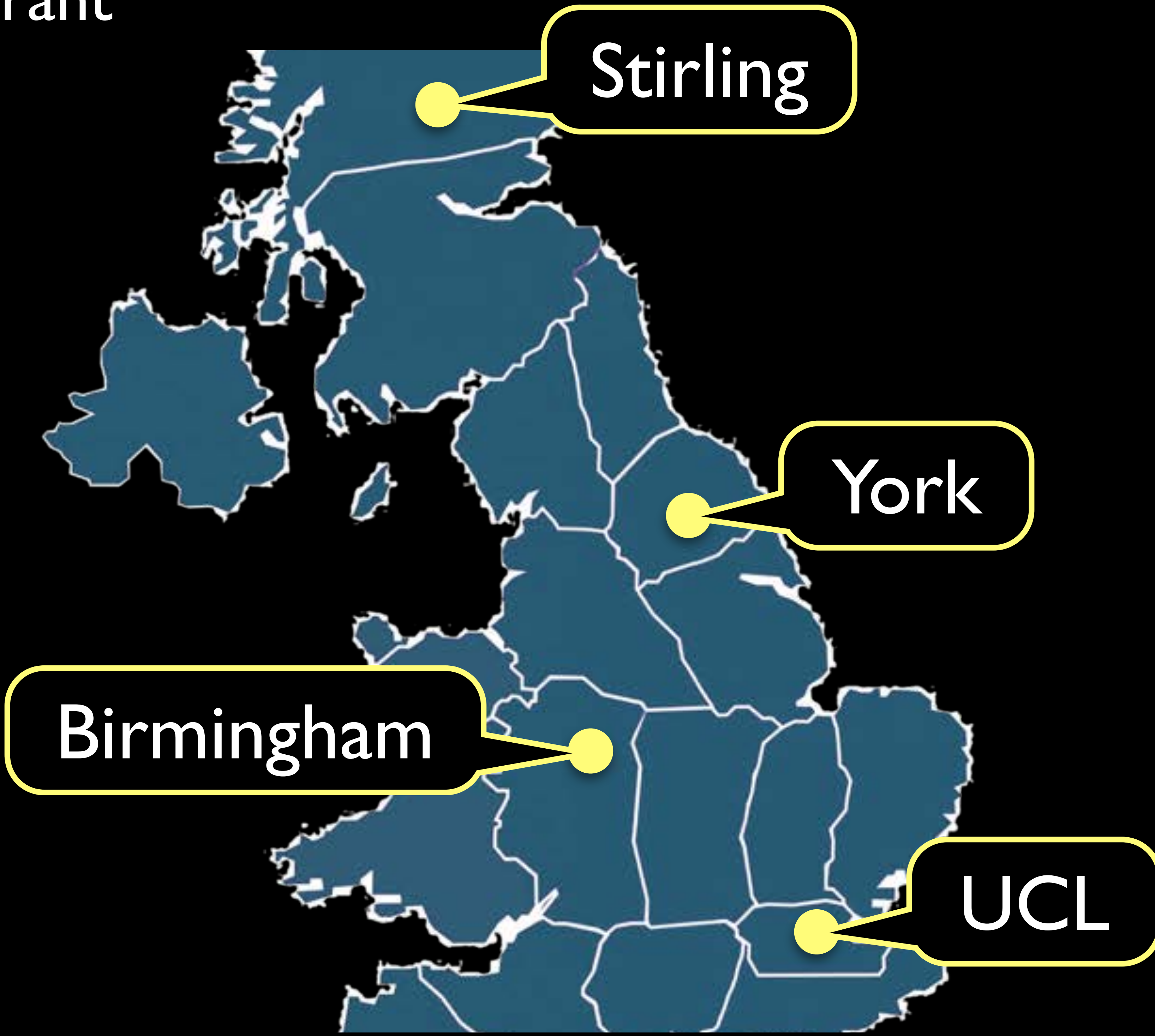
EPSRC
Grant



EPSRC Grant

EPSRC Grant

EPSRC Grant 2012-2018 PhD and post docs offered



Mark.Harman@ucl.ac.uk

SBSE Tutorial and Survey

SBSE Tutorial and Survey

Mark Harman, Phil McMinn, Jerffeson Teixeira de Souza and Shin Yoo. Search Based Software Engineering: Techniques, Taxonomy, Tutorial. Springer, 2012.

SBSE Tutorial and Survey

Mark Harman, Phil McMinn, Jerffeson Teixeira de Souza and Shin Yoo.
Search Based Software Engineering: Techniques, Taxonomy, Tutorial.
Springer, 2012.

Mark Harman, Afshin Mansouri and Yuanyuan Zhang.
Search Based Software Engineering: Trends, Techniques and Applications
ACM Computing Surveys.
45(1): Article 11, 2012.

SBSE Tutorial and Survey

Mark Harman, Phil McMinn, Jerffeson Teixeira de Souza and Shin Yoo.
Search Based Software Engineering: Techniques, Taxonomy, Tutorial.
Springer, 2012.

google: SBSE tutorial

Mark Harman, Afshin Mansouri and Yuanyuan Zhang.
Search Based Software Engineering: Trends, Techniques and Applications
ACM Computing Surveys.
45(1): Article 11, 2012.

SBSE Tutorial and Survey

Mark Harman, Phil McMinn, Jerffeson Teixeira de Souza and Shin Yoo.
Search Based Software Engineering: Techniques, Taxonomy, Tutorial.
Springer, 2012.

google: SBSE tutorial

Mark Harman, Afshin Mansouri and Yuanyuan Zhang.
Search Based Software Engineering: Trends, Techniques and Applications
ACM Computing Surveys.
45(1): Article 11, 2012.

google: SBSE survey

SBSE Tutorial and Survey

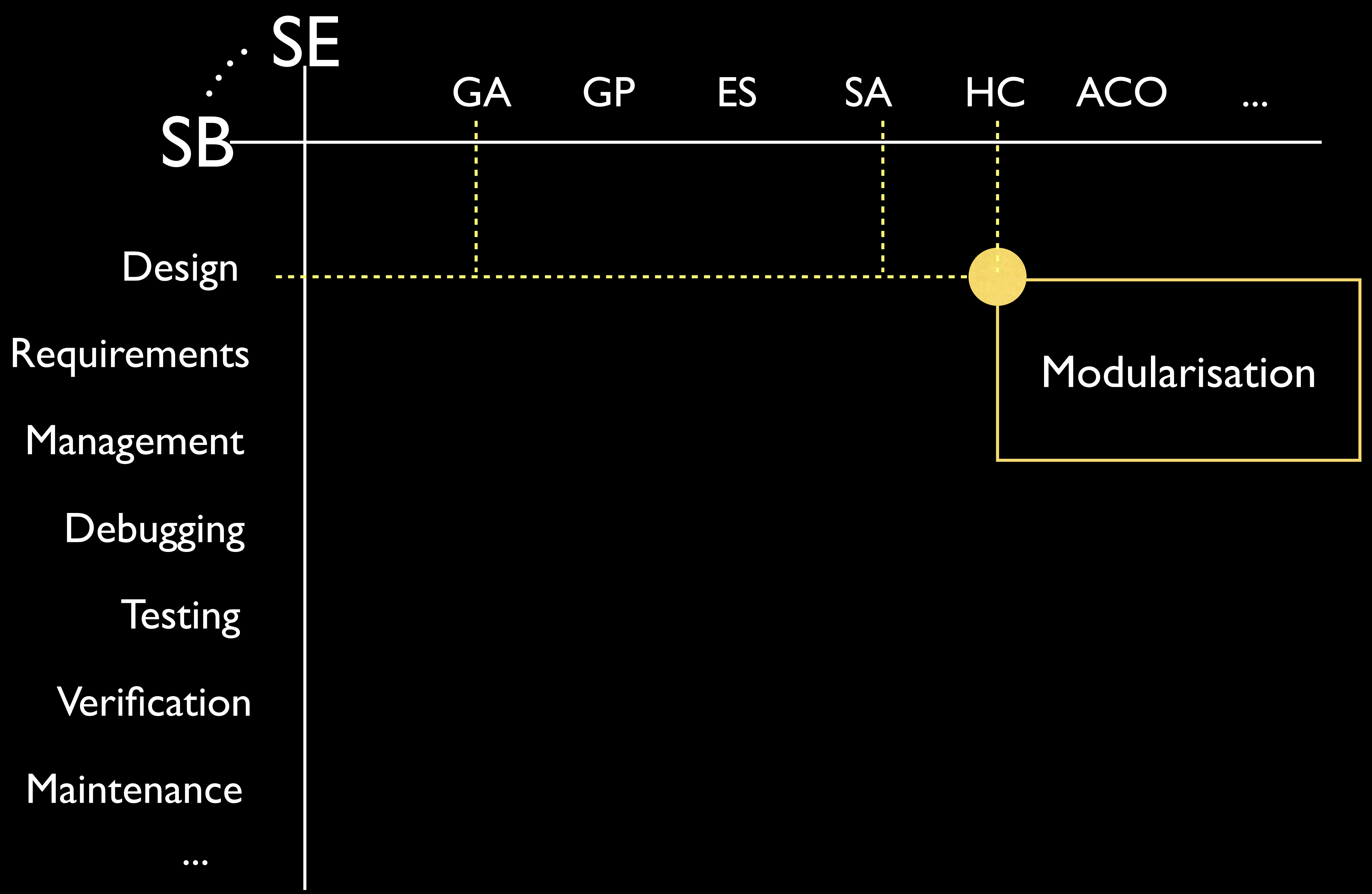
Mark Harman, Phil McMinn, Jerffeson Teixeira de Souza and Shin Yoo.
Search Based Software Engineering: Techniques, Taxonomy, Tutorial.
Springer, 2012.

google: SBSE tutorial

Mark Harman, Afshin Mansouri and Yuanyuan Zhang.
Search Based Software Engineering: Trends, Techniques and Applications
ACM Computing Surveys.
45(1): Article 11, 2012.

google: SBSE survey

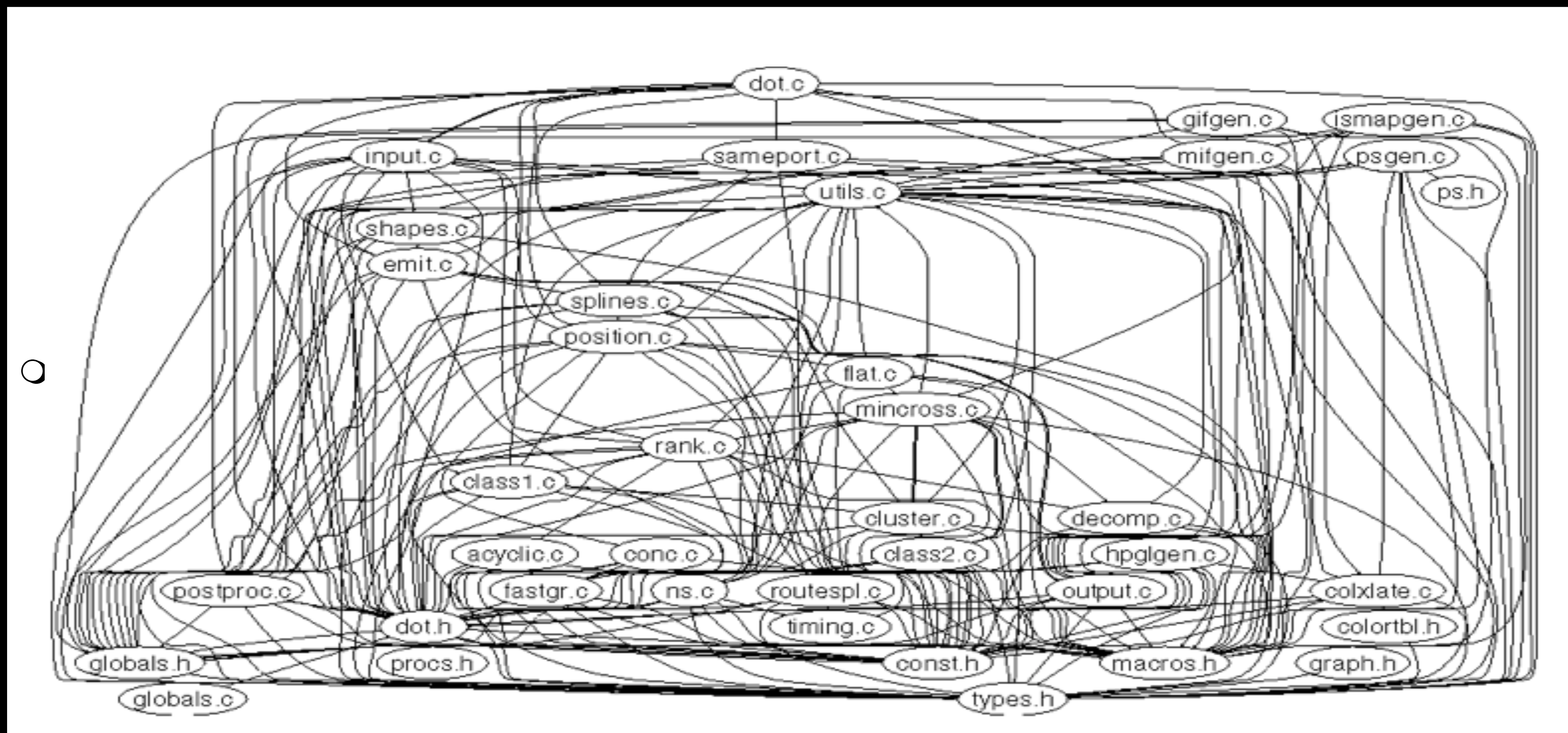
SB / SE	GA	GP	ES	SA	HC	ACO	...
Design							
Requirements							
Management							
Debugging							
Testing							
Verification							
Maintenance							
...							


SE
SB
GA
GP
ES
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Modularisation

Search based modularisation

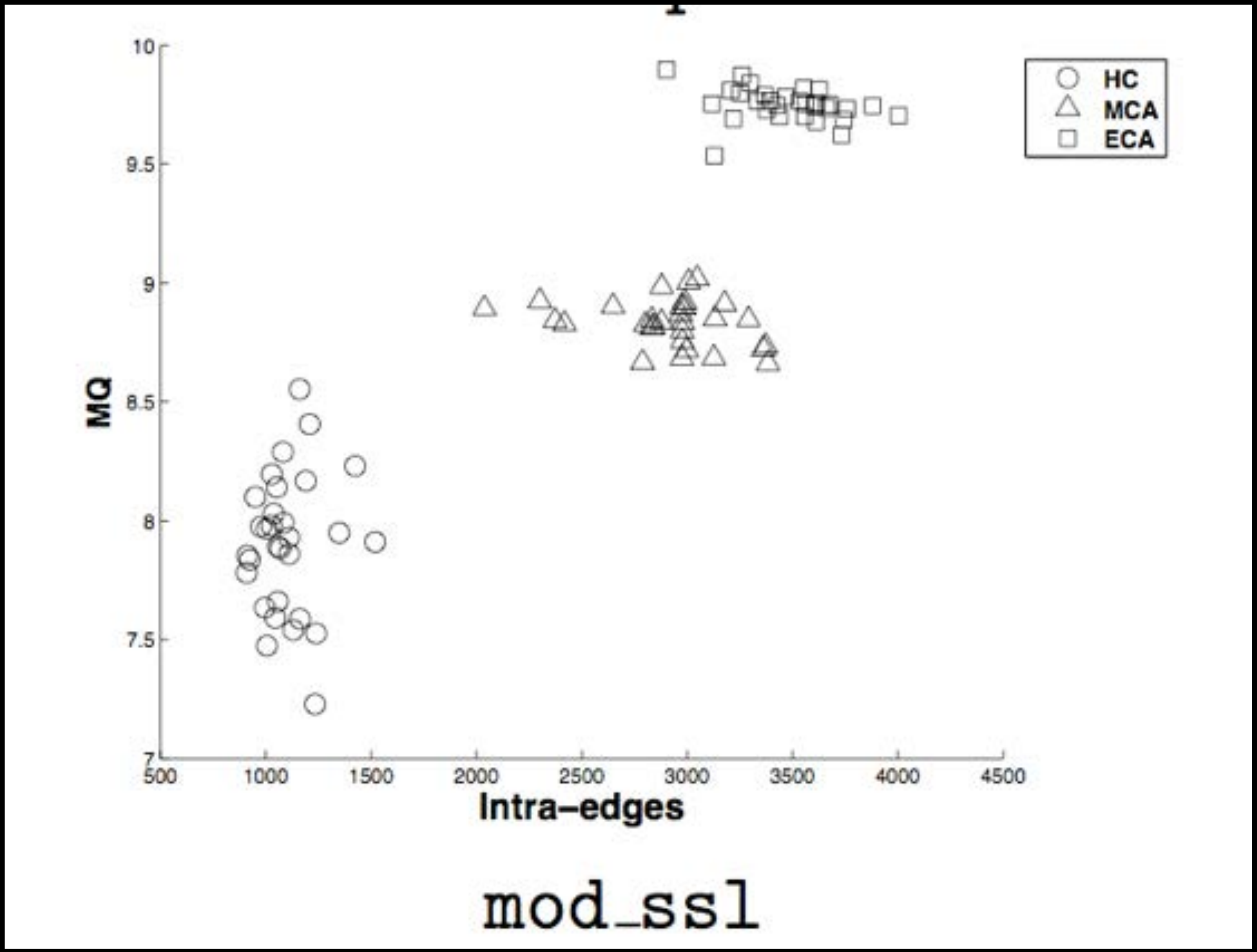
Mancoridis and Mitchell: IWPC 1998 and TSE 2006
Praditwong, Harman and Yao: TSE 2011
Barros: GECCO 2012

Search based modularisation



Mancoridis and Mitchell: IWPC 1998 and TSE 2006
Praditwong, Harman and Yao: TSE 2011
Barros: GECCO 2012

Search based modularisation



Mancoridis and Mitchell: IWPC 1998 and TSE 2006
Praditwong, Harman and Yao: TSE 2011
Barros: GECCO 2012

Take home: multi objective search helps even single objective problems

SE

SB

GA GP ES SA HC ACO ...

Design

Requirements

Management

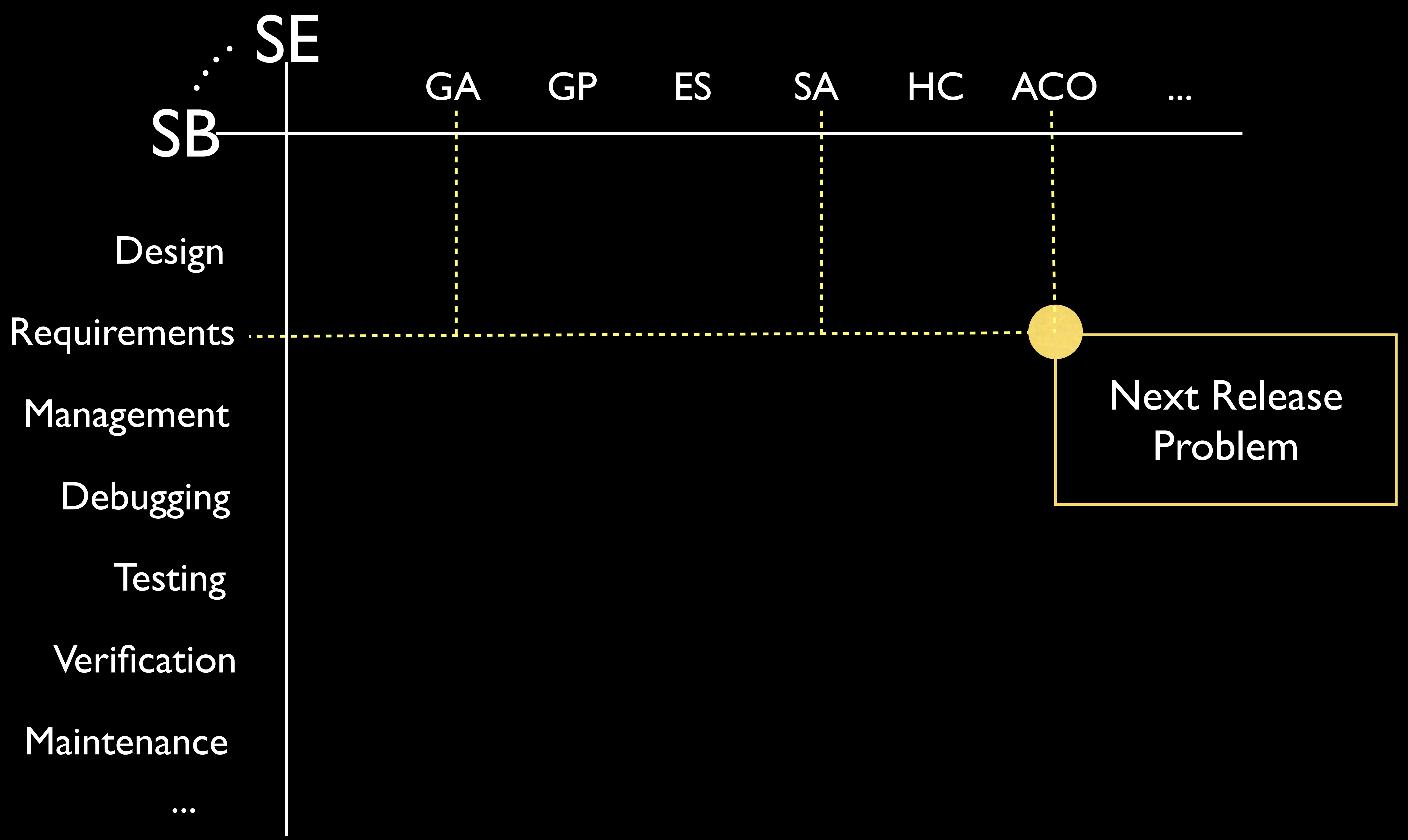
Debugging

Testing

Verification

Maintenance

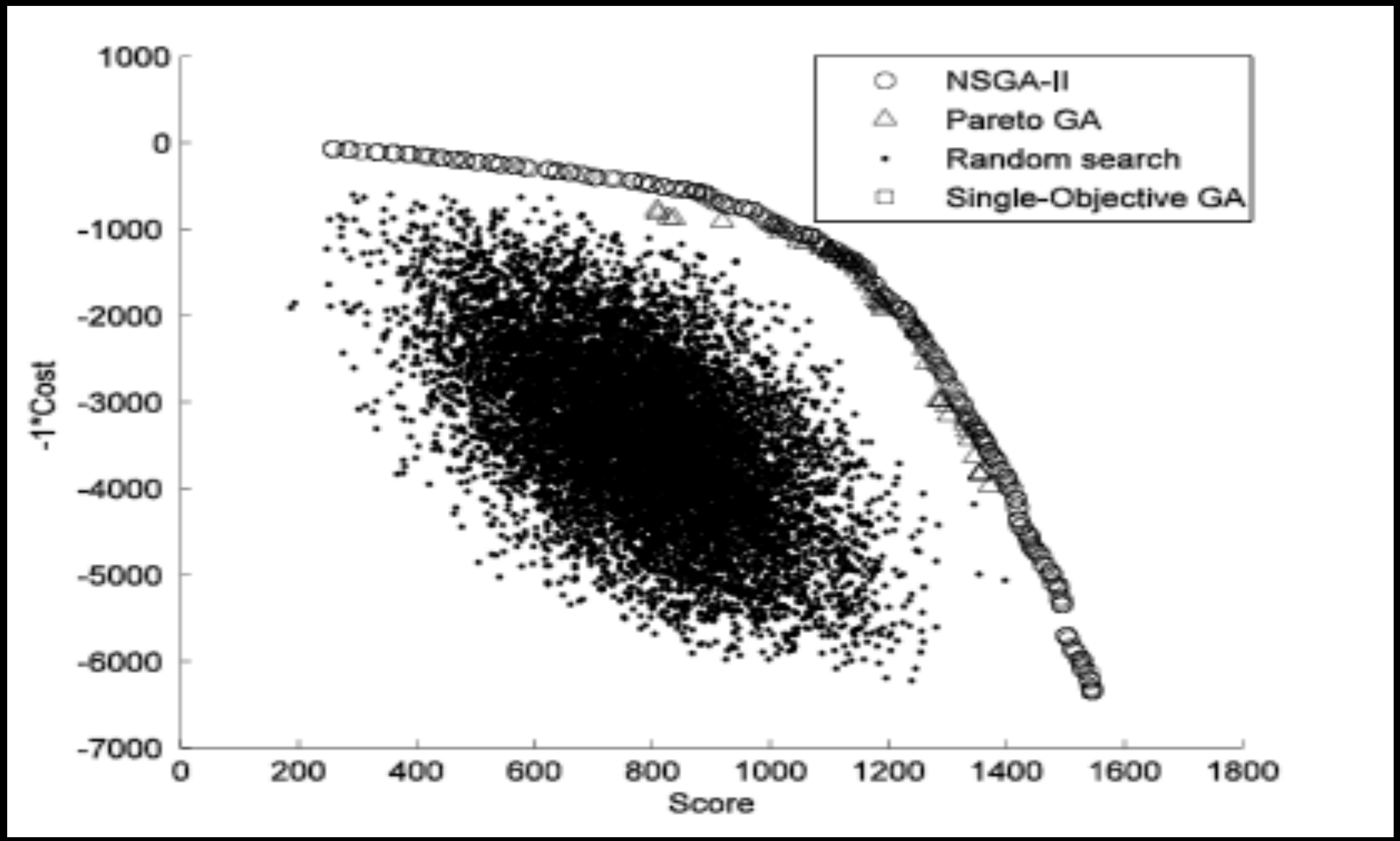
...


SE
SB
GA
GP
ES
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Next Release Problem

Search based requirements

Bagnall, Rayward-Smith and Whittley: IST 2001
Zhang, Harman and Mansouri: GECCO 2007
Saliu and Ruhe: FSE 2007

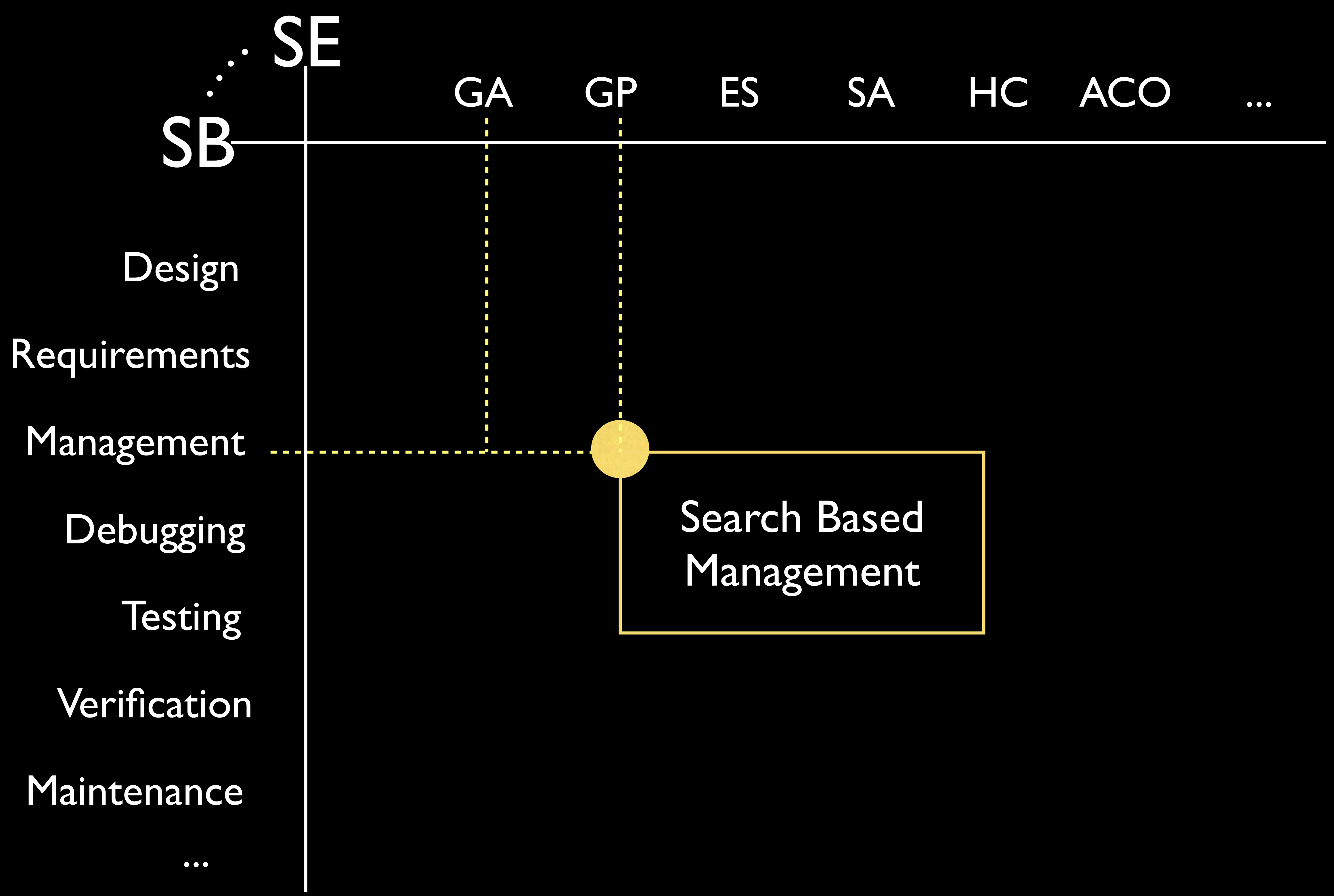
Search based requirements



Bagnall, Rayward-Smith and Whittley: IST 2001
Zhang, Harman and Mansouri: GECCO 2007
Saliu and Ruhe: FSE 2007

Take home: SBSE is well suited to cost value trade offs

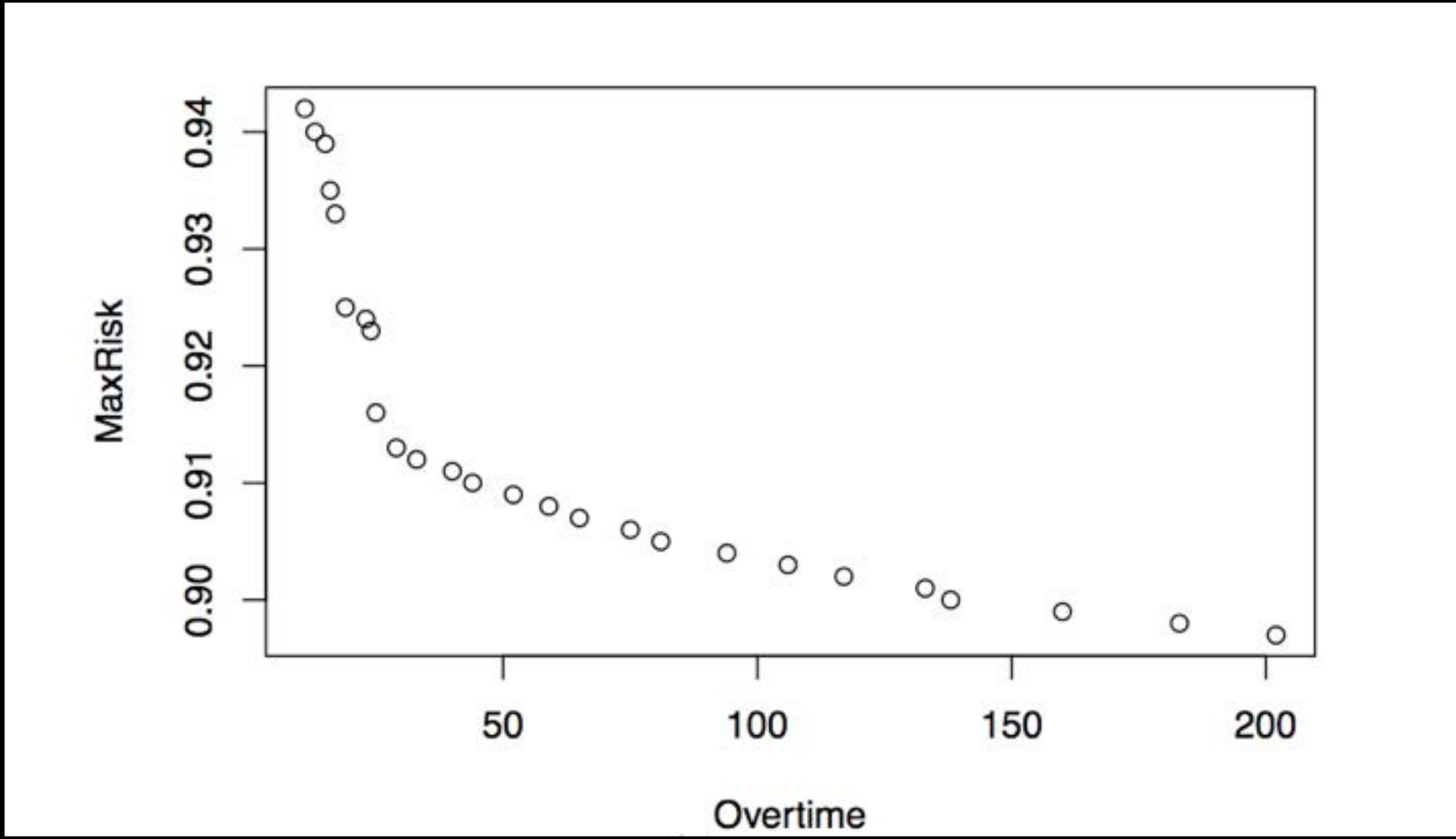
SB . . . SE	GA	GP	ES	SA	HC	ACO	...
Design							
Requirements							
Management							
Debugging							
Testing							
Verification							
Maintenance							
...							


SE
SB
GA
GP
ES
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Search Based Management

Search based management

Dolado: IST 2001
Chicano and Alba: MIC 2005
Ferrucci, Harman, Ren and Sarro: ICSE 2013

Search based management



Dolado: IST 2001
Chicano and Alba: MIC 2005
Ferrucci, Harman, Ren and Sarro: ICSE 2013

Take home: SBSE can help analyse risk - reward

SE

SB

GA GP ES SA HC ACO ...

Design

Requirements

Management

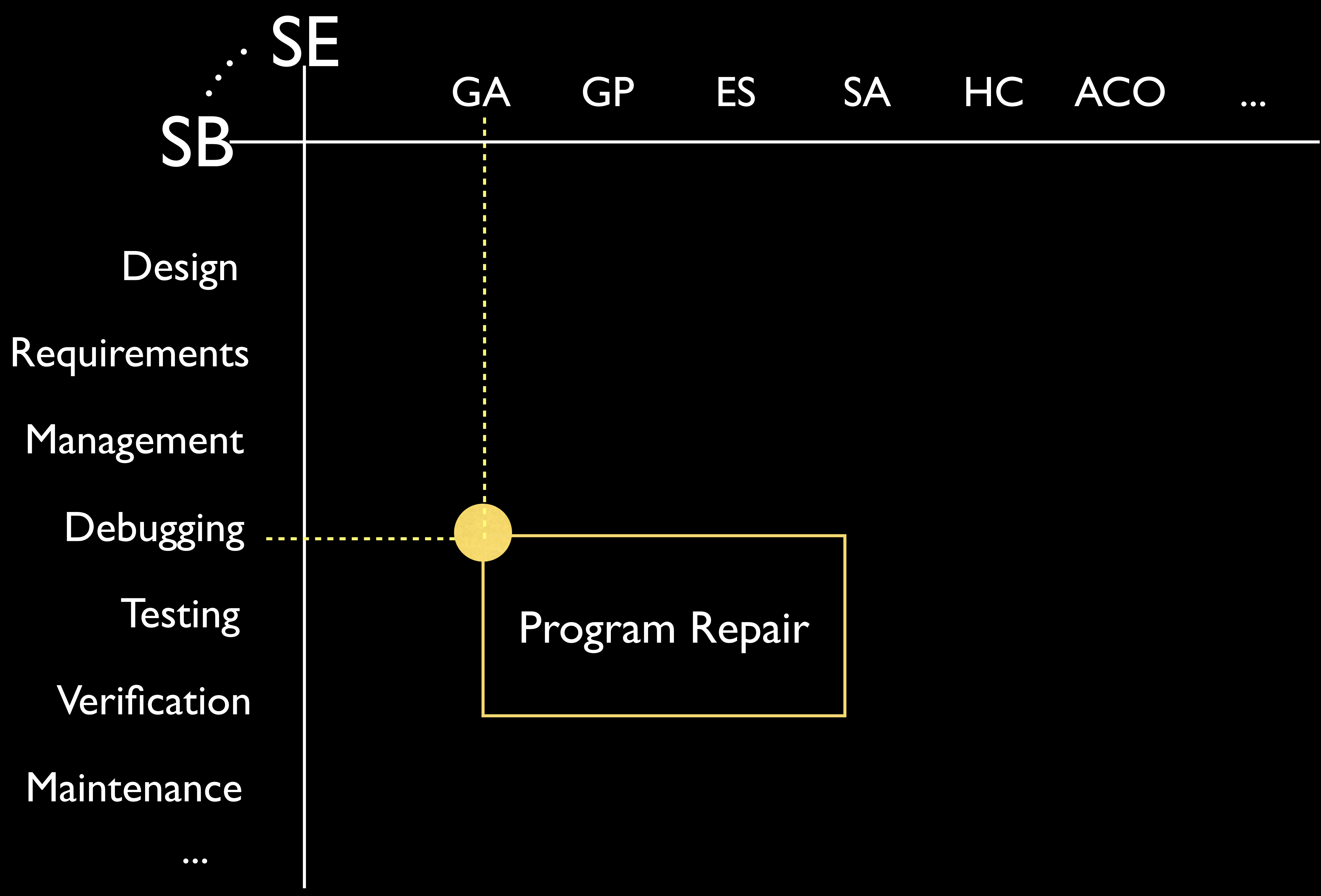
Debugging

Testing

Verification

Maintenance

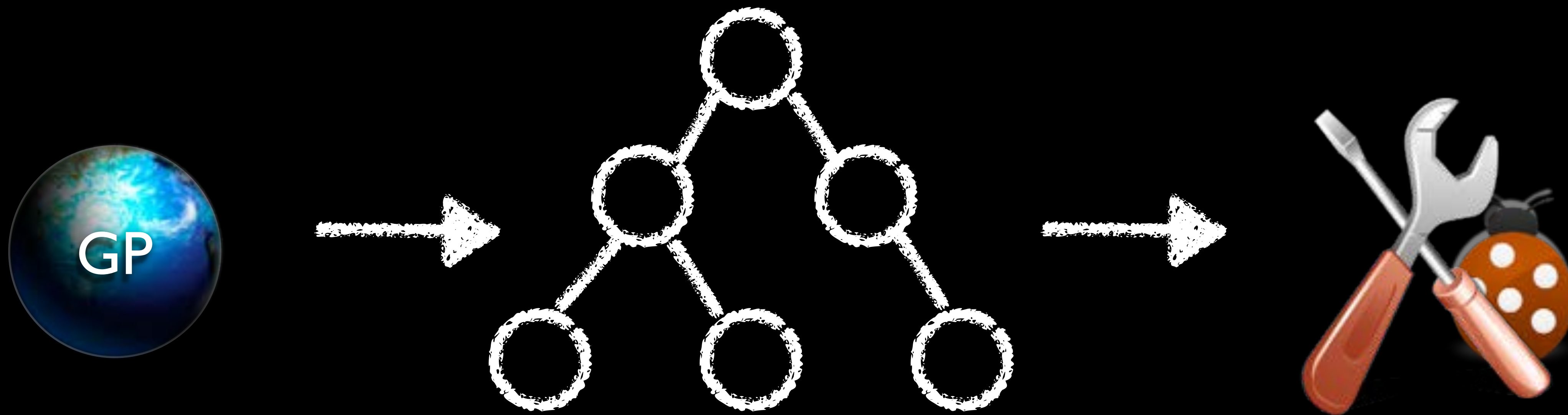
...


SE
SB
GA
GP
ES
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Program Repair

Search based repair

Arcuri and Yao: CEC 2008
Weimer, Nguyen, Le Goues and Forrest: ICSE 2009
Kim, Nam, Song and Kim: ICSE 2013

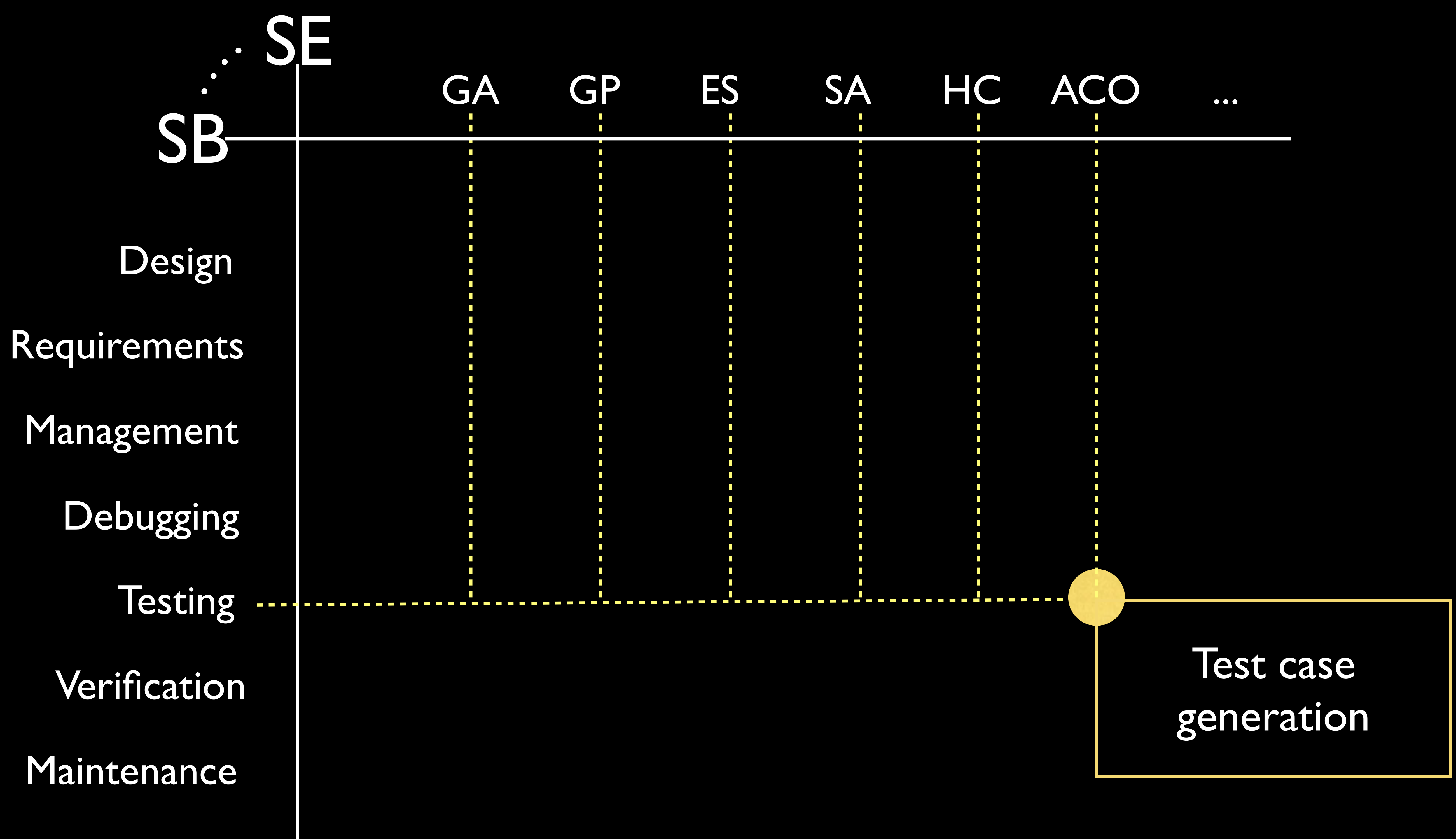
Search based repair



Arcuri and Yao: CEC 2008
Weimer, Nguyen, Le Goues and Forrest: ICSE 2009
Kim, Nam, Song and Kim: ICSE 2013

Take home: some (~50%) real bugs are easy to fix

SB . . . SE	GA	GP	ES	SA	HC	ACO	...
Design							
Requirements							
Management							
Debugging							
Testing							
Verification							
Maintenance							
...							

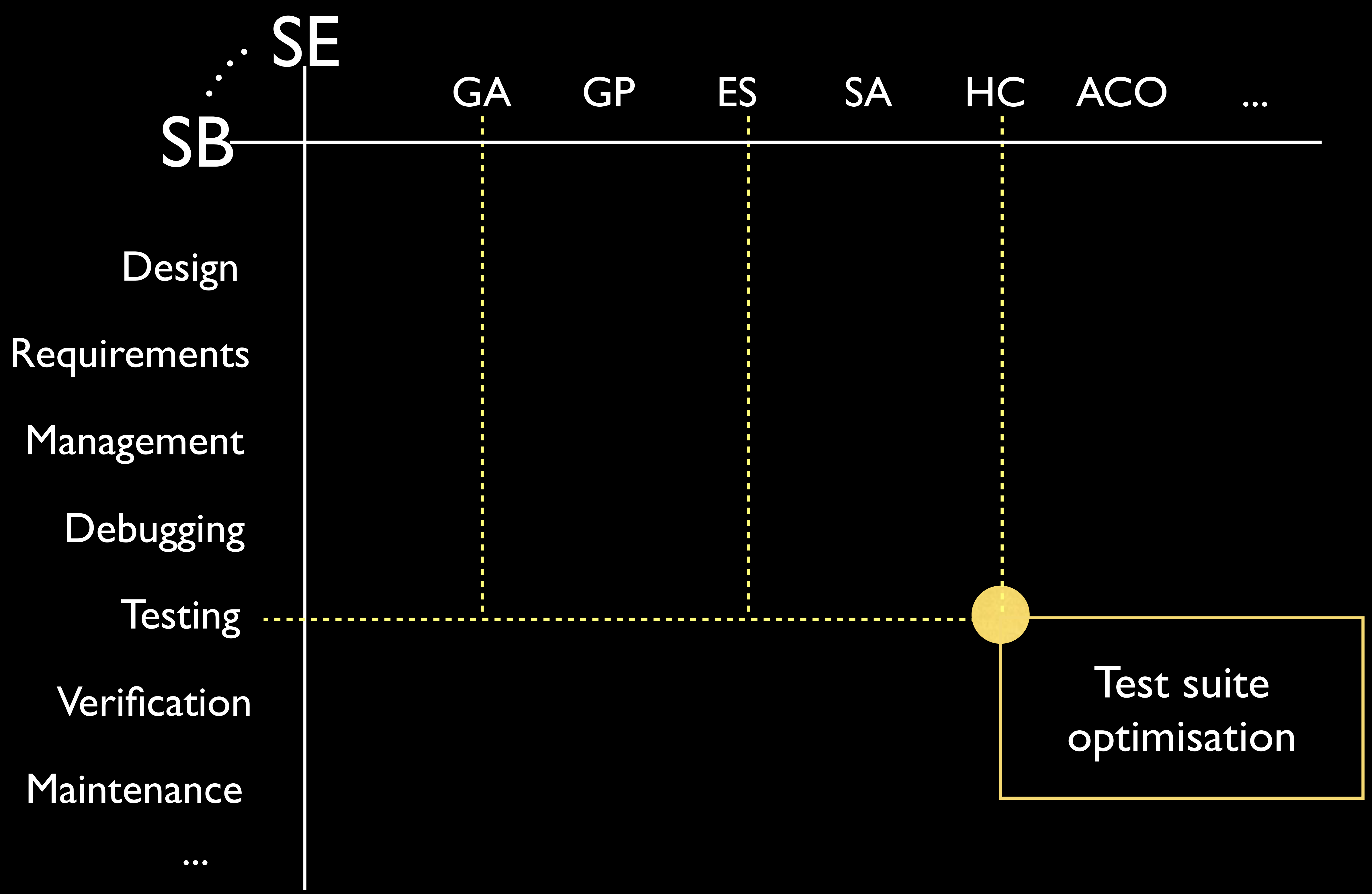


Take home: SBSE *loves* testing problem: adequacy = fitness

SE

SB

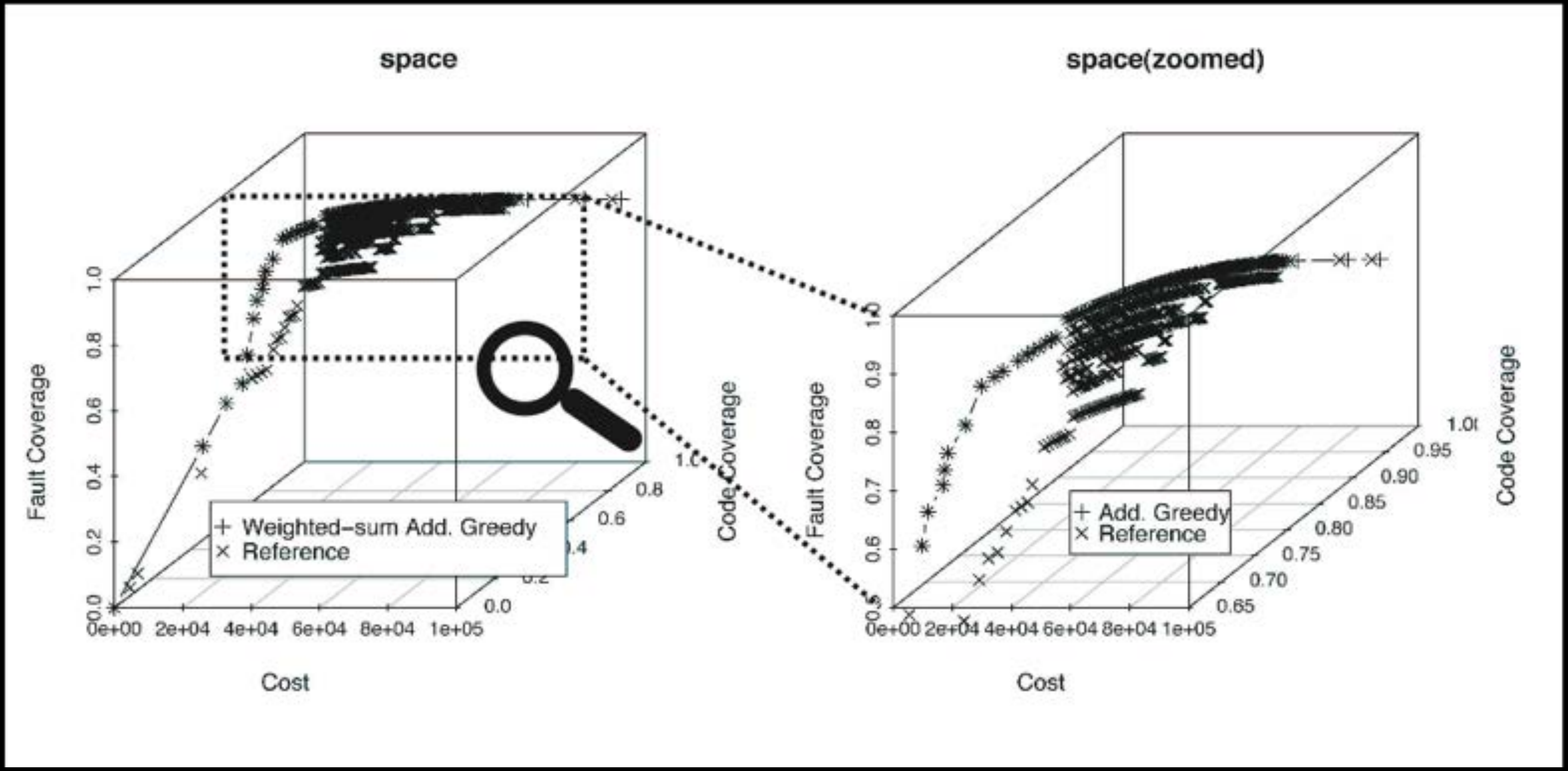
	GA	GP	ES	SA	HC	ACO	...
Design							
Requirements							
Management							
Debugging							
Testing							
Verification							
Maintenance							
...							


SE
SB
GA
GP
ES
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Test suite optimisation

Test suite optimisation

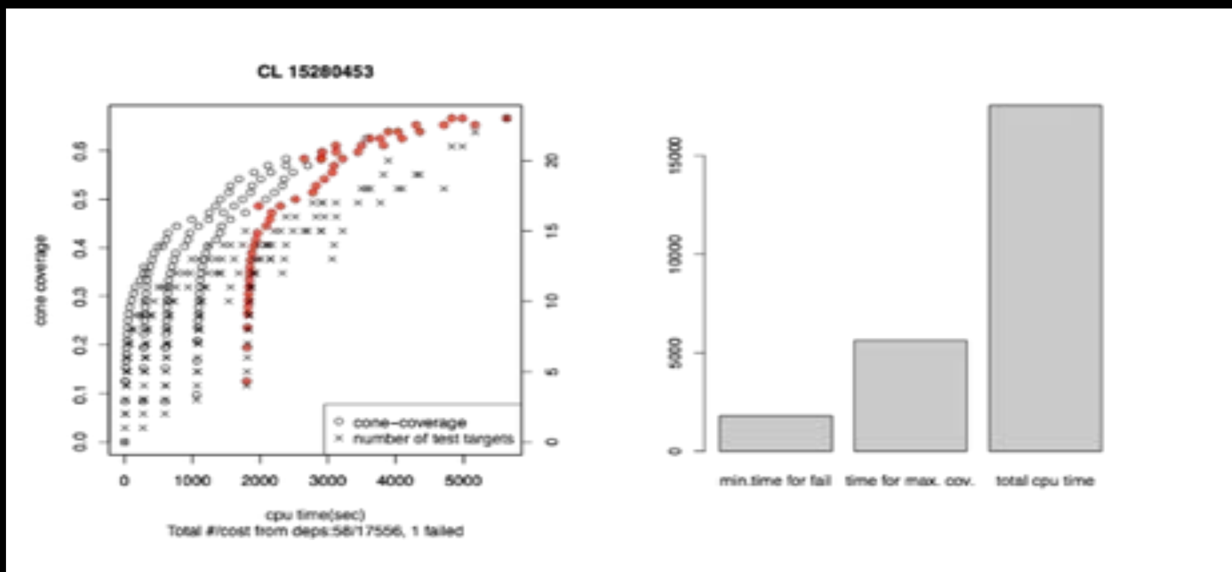
Yoo and Harman: ISSTA 2007
Li, Harman and Hierons: TSE 2007
Mirarab, Akhlaghi and Tahvildari: TSE 2012

Test suite selection

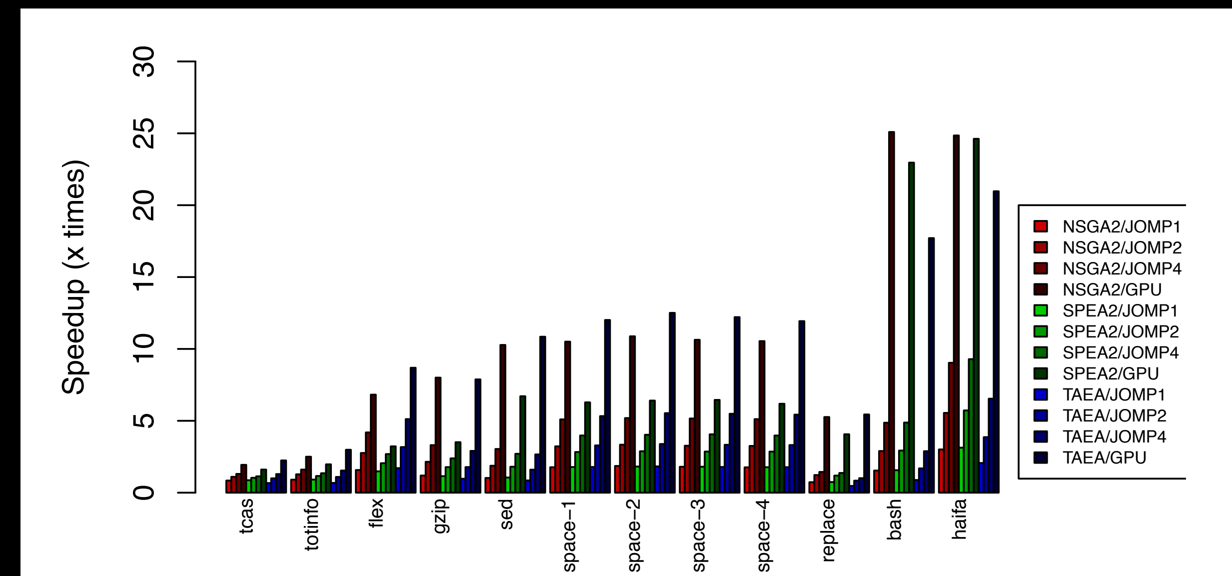


Yoo and Harman: ISSTA 2007
Li, Harman and Hierons: TSE 2007
Mirarab, Akhlaghi and Tahvildari: TSE 2012

Test suite selection



Google Yoo, Nilsson and Harman, FSE 2011



IBM Yoo, Harman and Ur, EMSE 2013

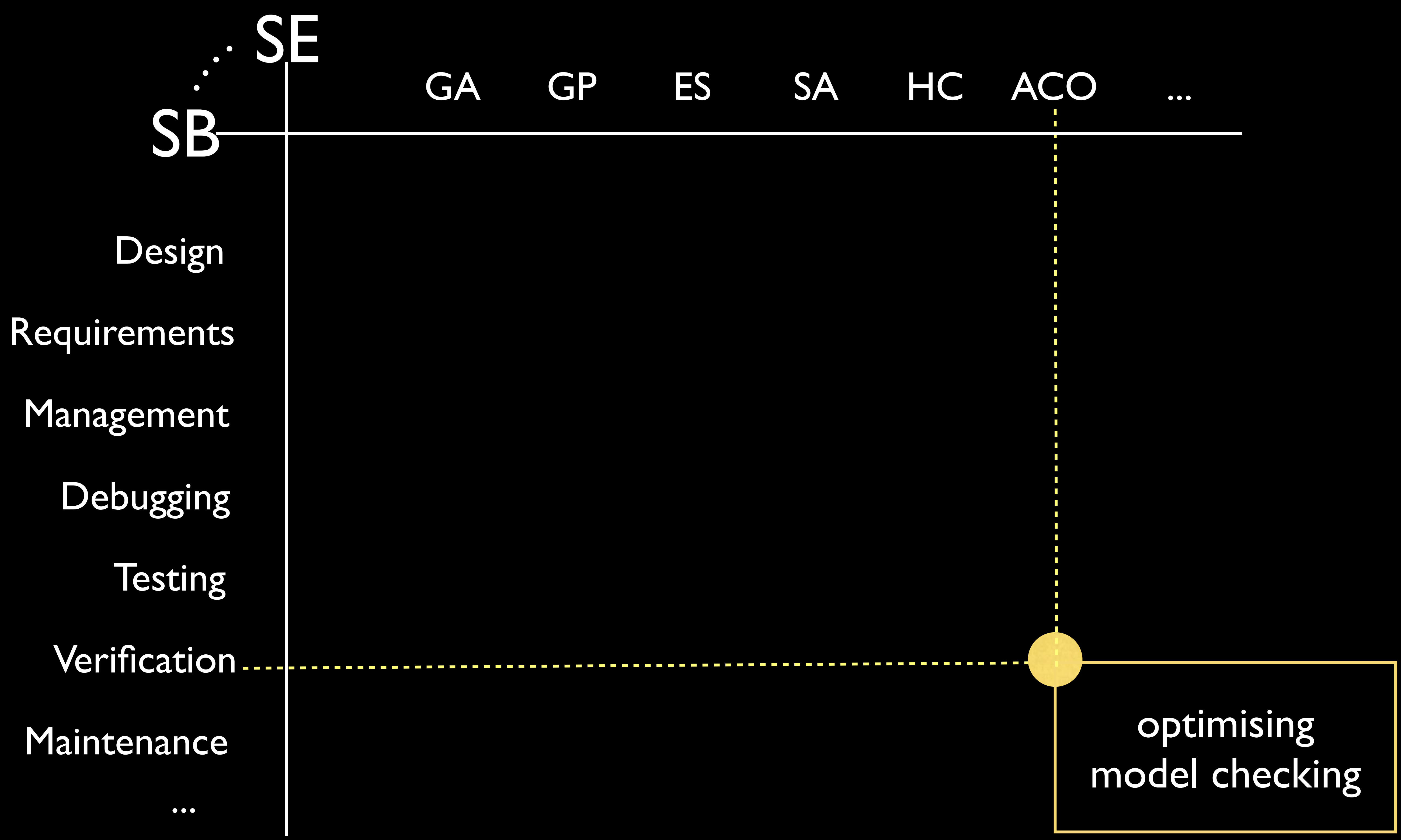
Find faults faster

Optimise faster using GPGPU

Yoo and Harman: ISSTA 2007
Li, Harman and Hierons: TSE 2007
Mirarab, Akhlaghi and Tahvildari: TSE 2012

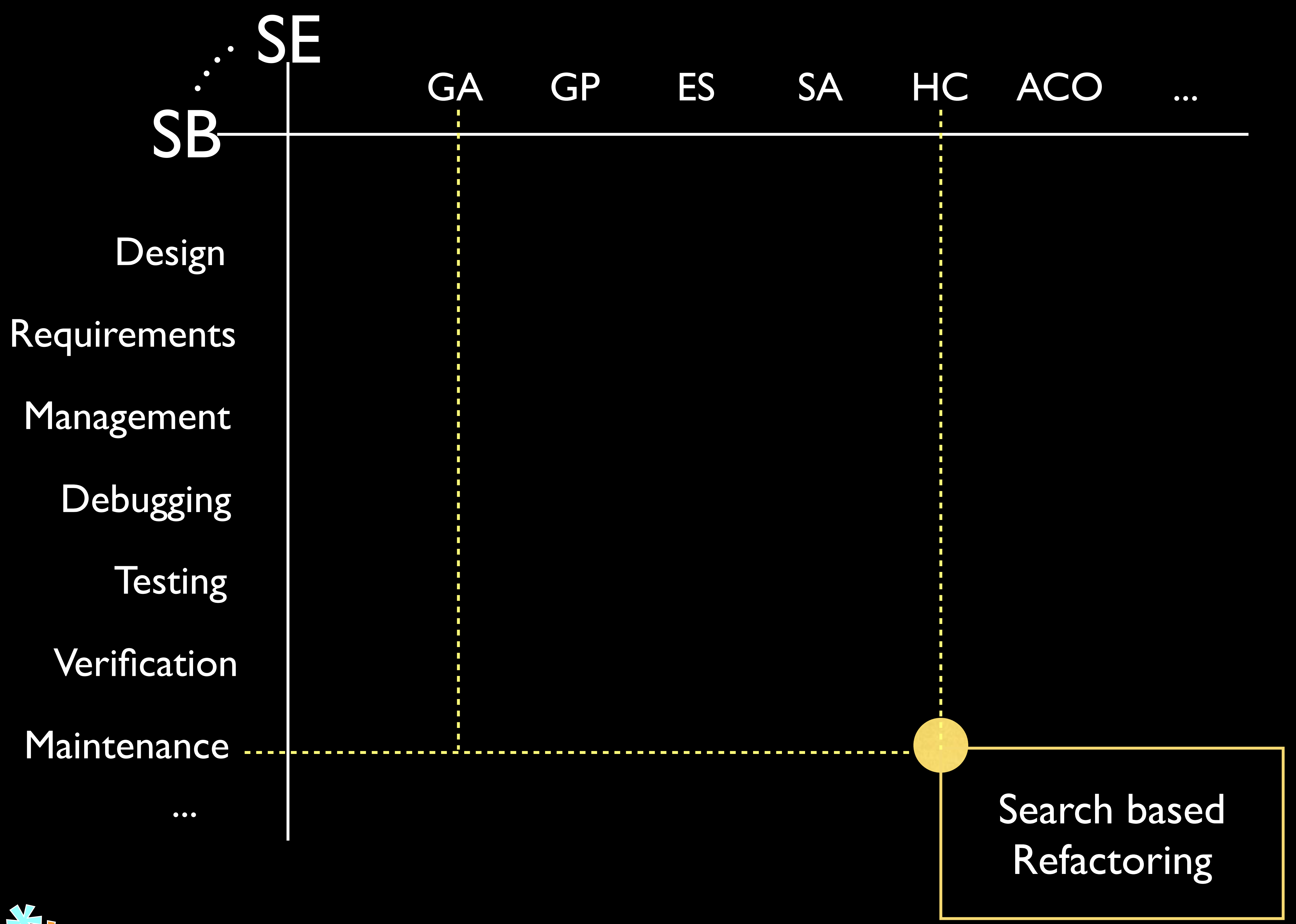
Take home: regression testing is *all about optimisation*

SB . . . SE	GA	GP	ES	SA	HC	ACO	...
Design							
Requirements							
Management							
Debugging							
Testing							
Verification							
Maintenance							
...							



Take home: Model checkers search large spaces

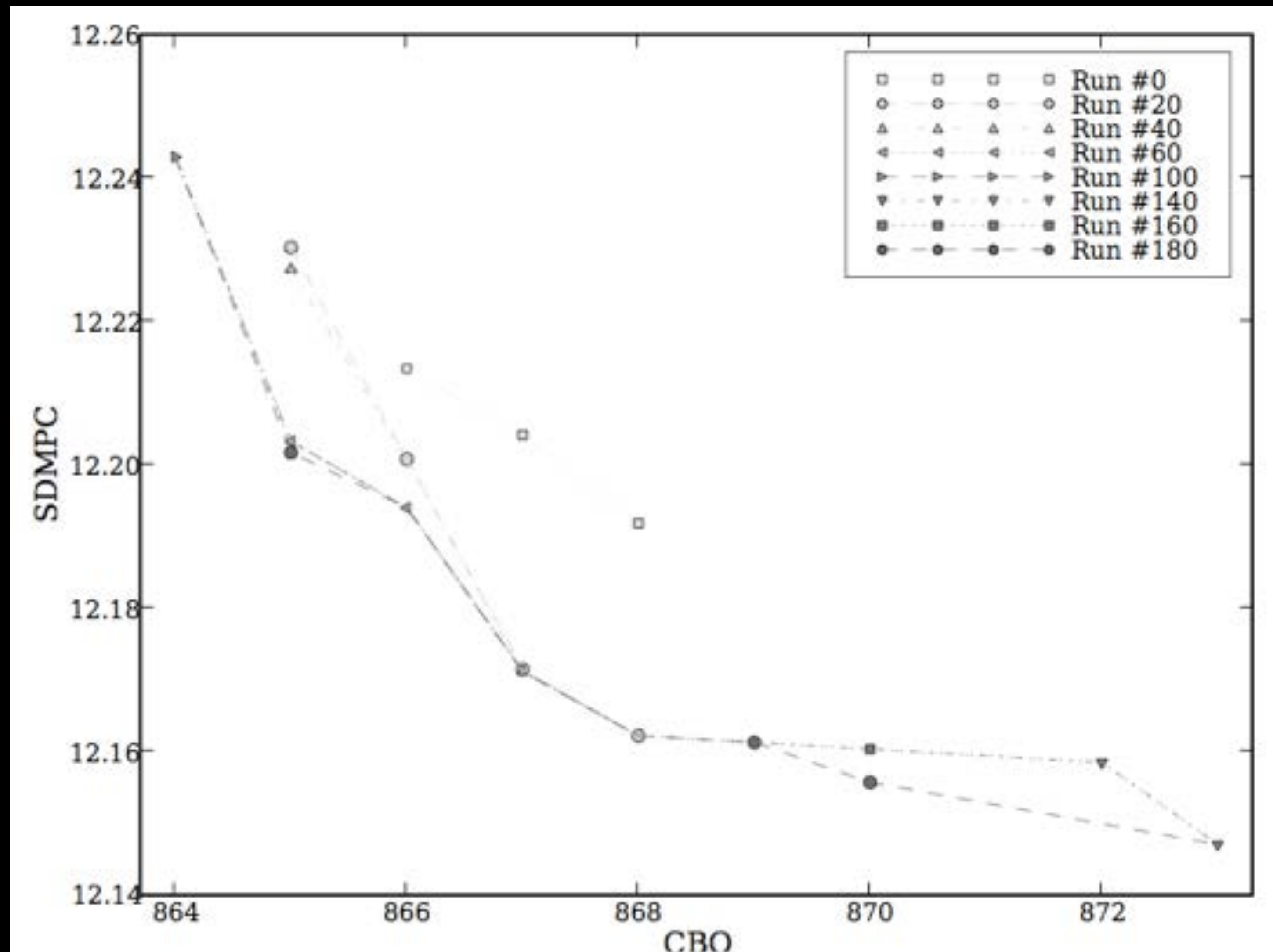
SB ··· SE	GA	GP	ES	SA	HC	ACO	...
Design							
Requirements							
Management							
Debugging							
Testing							
Verification							
Maintenance							
...							


SE
SB
GA
GP
ES
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Search based Refactoring

Search based refectoring

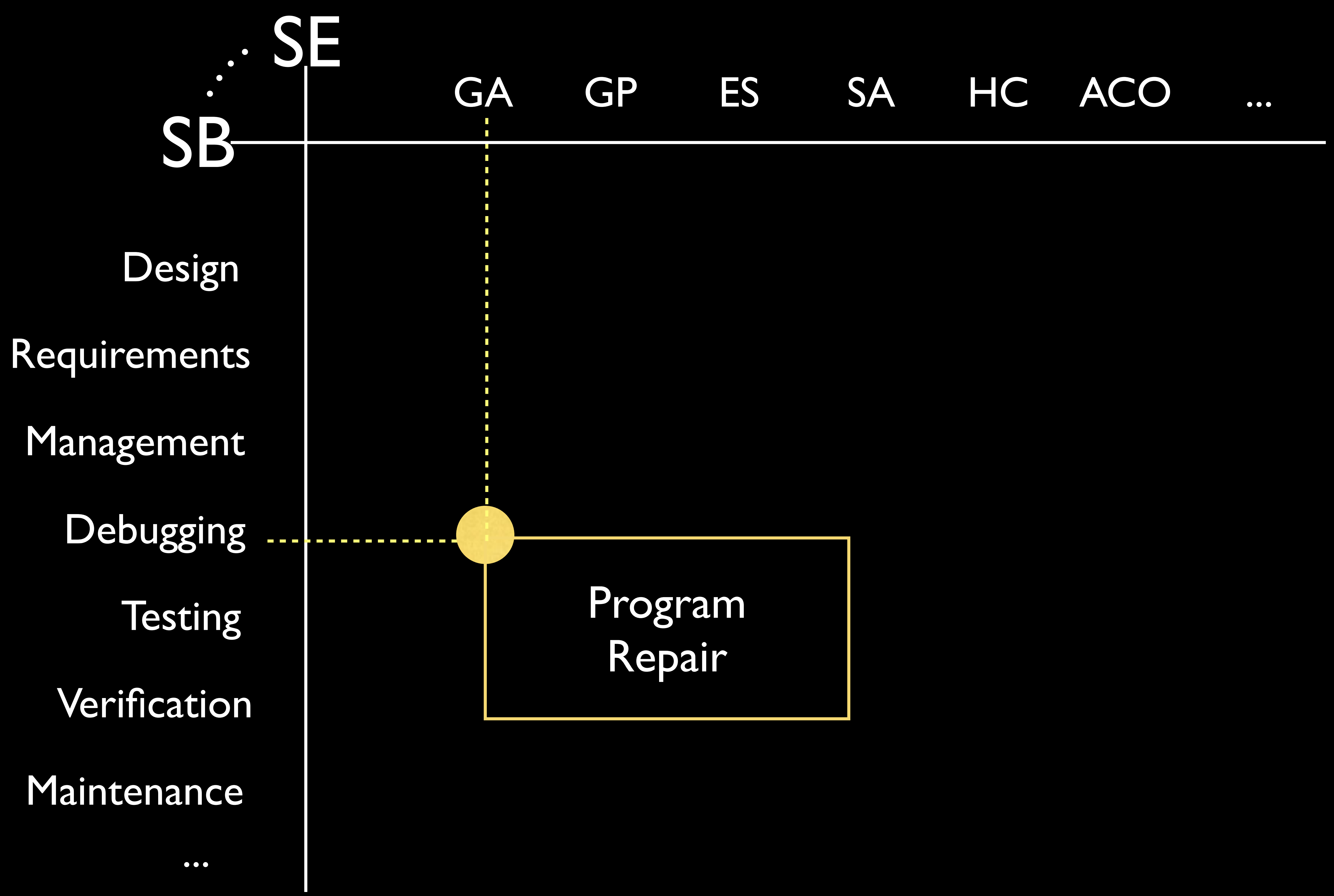
O'Keeffe and Ó Cinnéide: CSMR 2003
Harman and Tratt: GECCO 2007
Jensen and Cheng: GECCO 2010

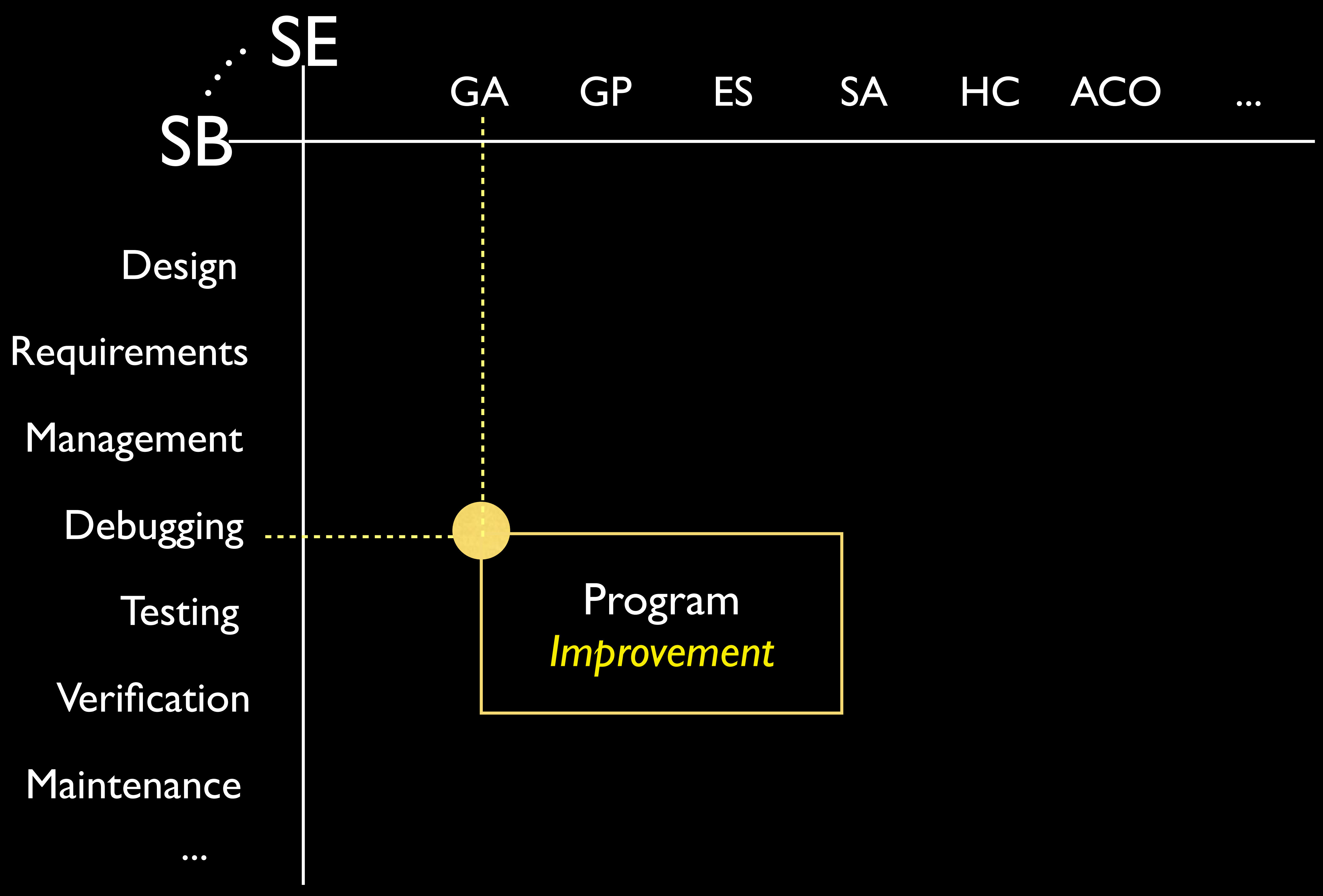
Search based refectoring



O'Keeffe and Ó Cinnéide: CSMR 2003
Harman and Tratt: GECCO 2007
Jensen and Cheng: GECCO 2010

Take home: refactoring is a multi objective optimisation problem


SE
SB
GA
GP
ES
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Program Repair


SE
SB
GA
GP
ES
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Program
Improvement

GISMOE

Pareto program surface is automatically constructed to support dialog with the software designer concerning trade offs in the solution space of programs

what is a pareto program surface?

There is a paper to accompany this keynote

The GISMOE challenge: Constructing the Pareto Program Surface Using Genetic Programming to Find Better Programs*

Mark Harman[1], William B. Langdon[1], Yue Jia[1], David R. White[2], Andrea Arcuri[3], John A. Clark[4]
[1]CREST Centre, University College London, Gower Street, London, WC1E 6BT, UK.
[2]School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, Scotland, UK.
[3]Simula Research Laboratory, P. O. Box 134, 1325 Lysaker, Norway.
[4]Department of Computer Science, University of York, Deramore Lane, York, YO10 5GH, UK.

ABSTRACT

Optimising programs for non-functional properties such as speed, size, throughput, power consumption and bandwidth can be demanding; pity the poor programmer who is asked to cater for them all at once! We set out an alternate vision for a new kind of software development environment inspired by recent results from Search Based Software Engineering (SBSE). Given an input program that satisfies the functional requirements, the proposed programming environment will automatically generate a set of candidate program implementations, all of which share functionality, but each of which differ in their non-functional trade offs. The software designer navigates this diverse Pareto surface of candidate implementations, gaining insight into the trade offs and selecting solutions for different platforms and environments, thereby stretching beyond the reach of current compiler technologies. Rather than having to focus on the details required to manage complex, inter-related and conflicting, non-functional trade offs, the designer is thus freed to explore, to understand, to control and to decide rather than to construct.

Categories and Subject Descriptors

D.2 [**Software Engineering**]

General Terms

Algorithms, Design, Experimentation, Human Factors, Languages, Measurement, Performance, Verification.

*This position paper accompanies the keynote given by Mark Harman's at the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 12) in Essen, Germany. It is joint work with Bill Langdon, Yue Jia, David White, Andrea Arcuri and John Clark, funded by the EPSRC grants SEBASE (EP/D050863, EP/D050618 and EP/D052785), GISMO (EP/I033688) and DAASE (EP/J017515/) and by EU project FITTEST (257574).

ASE'12, September 3–7, 2012, Essen, Germany.

Keywords

SBSE, Search Based Optimization, Compilation, Non-functional Properties, Genetic Programming, Pareto Surface.

1. INTRODUCTION

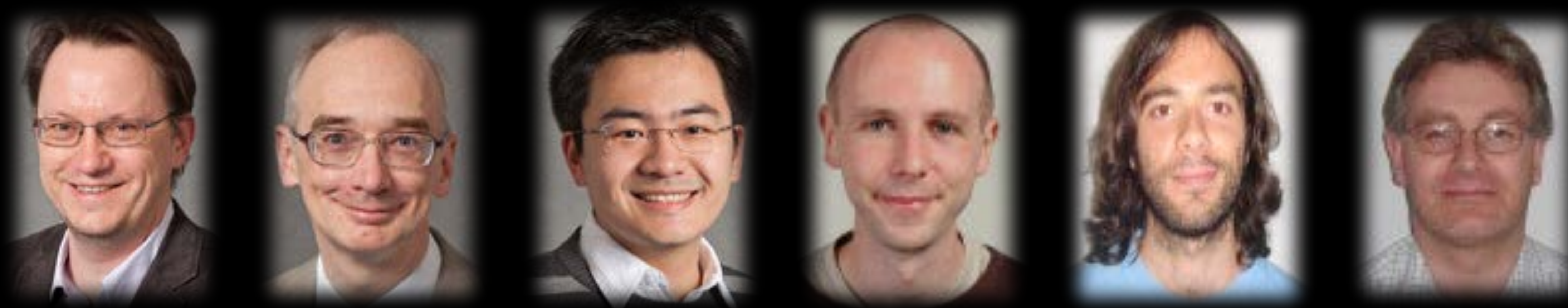
Humans find it hard to develop systems that balance many competing and conflicting non-functional objectives. Even meeting a single objective, such as execution time, requires automated support in the form of compiler optimisation. However, though most compilers can optimise compiled code for both speed and size, the programmer may find themselves making arbitrary choices when such objective are in conflict with one another.

Furthermore, speed and size are but two of many objectives that the next generation of software systems will have to consider. There are many others such as bandwidth, throughput, response time, memory consumption and resource access. It is unrealistic to expect an engineer to decide, up front, on the precise weighting that they attribute to each such non-functional property, nor for the engineer even to know what might be achievable in some unfamiliar environment in which the system may be deployed.

Emergent computing application paradigms require systems that are not only reliable, compact and fast, but which also optimise many different competing and conflicting objectives such as response time, throughput and consumption of resources (such as power, bandwidth and memory). As a result, operational objectives (the so-called non-functional properties of the system) are becoming increasingly important and uppermost in the minds of software engineers.

Human software developers cannot be expected to optimally balance these multiple competing constraints and may miss potentially valuable solutions should they attempt to do so. Why should they have to? How can a programmer assess (at code writing time) the behaviour of their code with regard to non-functional properties on a platform that may not yet have been built?

To address this conundrum we propose a development environment that distinguishes between functional and non-functional properties. In this environment, the functional properties remain the preserve of the human designer, while the optimisation of non-functional properties is left to the machine. That is, the *choice* of the non-functional properties to be considered will remain a decision for the human software designer.



The GISMOE challenge: Constructing the Pareto Program Surface Using Genetic Programming to Find Better Programs

Mark Harman[1], William B. Langdon[1], Yue Jia[1], David R. White[2], Andrea Arcuri[3], John A. Clark[4]
[1]CREST Centre, University College London, Gower Street, London, WC1E 6BT, UK.
[2]School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, Scotland, UK.
[3]Simula Research Laboratory, P. O. Box 134, 1325 Lysaker, Norway.
[4]Department of Computer Science, University of York, Deramore Lane, York, YO10 5GH, UK.

ABSTRACT

Optimising programs for non-functional properties such as speed, size, throughput, power consumption and bandwidth can be demanding; pity the poor programmer who is asked to cater for them all at once! We set out an alternate vision for a new kind of software development environment inspired by recent results from Search Based Software Engineering (SBSE). Given an input program that satisfies the functional requirements, the proposed programming environment will automatically generate a set of candidate program implementations, all of which share functionality, but each of which differ in their non-functional trade offs. The software designer navigates this diverse Pareto surface of candidate implementations, gaining insight into the trade offs and selecting solutions for different platforms and environments, thereby stretching beyond the reach of current

Keywords

SBSE, Search Based Optimization, Compilation, Non-functional Properties, Genetic Programming, Pareto Surface.

1. INTRODUCTION

Humans find it hard to develop systems that balance many competing and conflicting non-functional objectives. Even meeting a single objective, such as execution time, requires automated support in the form of compiler optimisation. However, though most compilers can optimise compiled code for both speed and size, the programmer may find themselves making arbitrary choices when such objective are in conflict with one another.

Furthermore, speed and size are but two of many objectives that the next generation of software systems will have *programming is changing*

Functional Requirements

Non-Functional Requirements

Requirements

Functional Requirements

functionality of the Program

Non-Functional Requirements

Execution Time

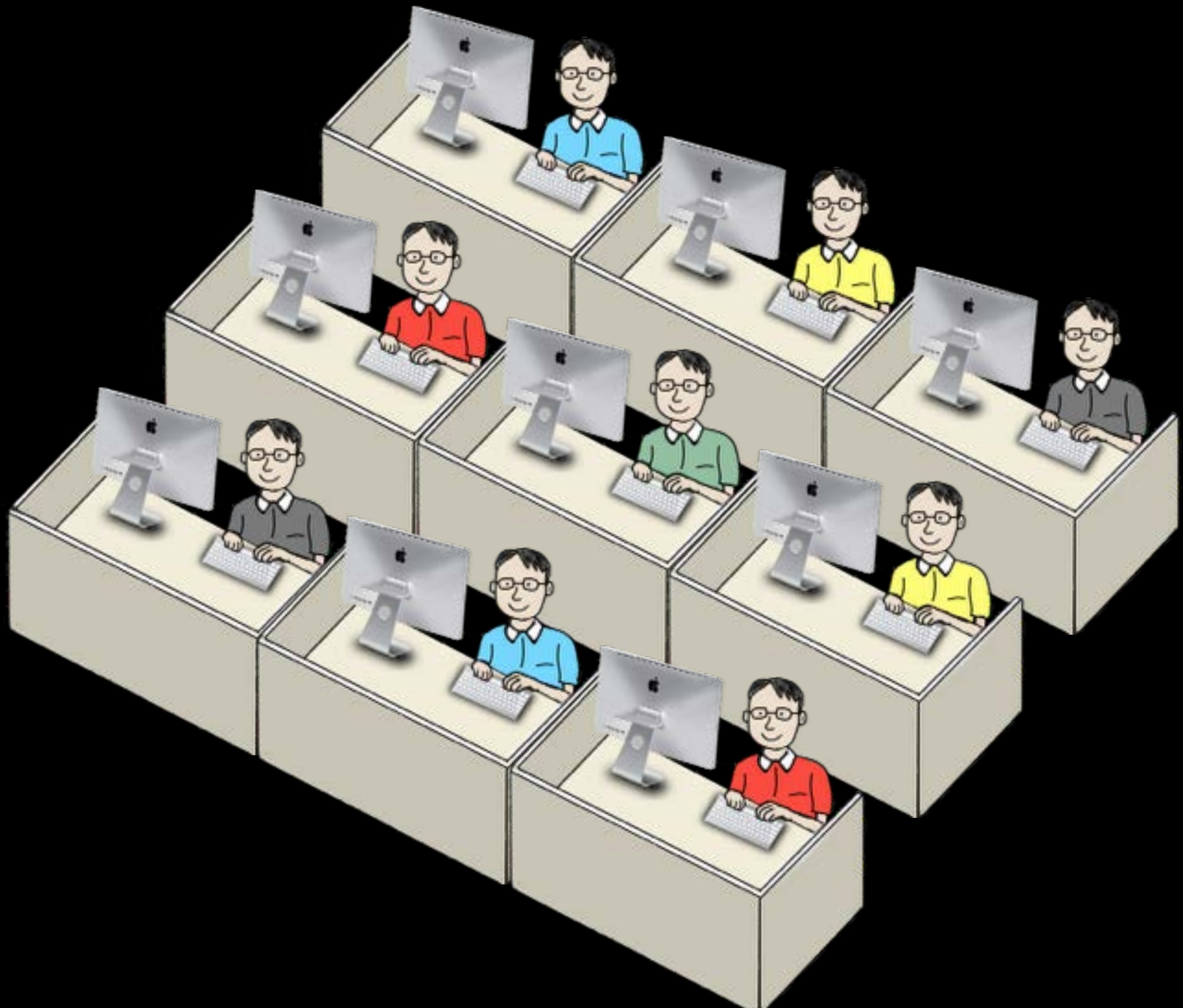
Memory

Bandwidth

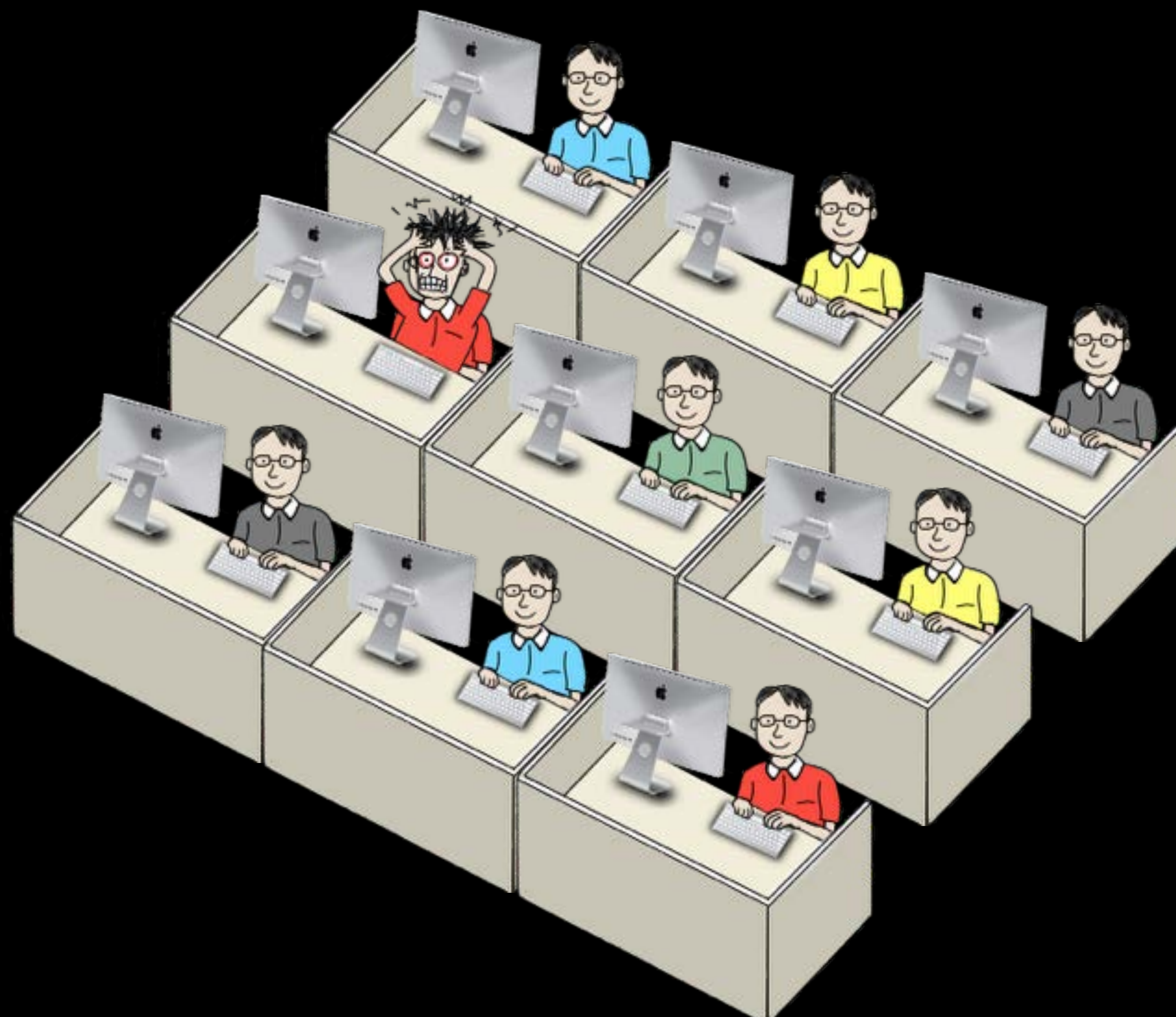
Battery

Size

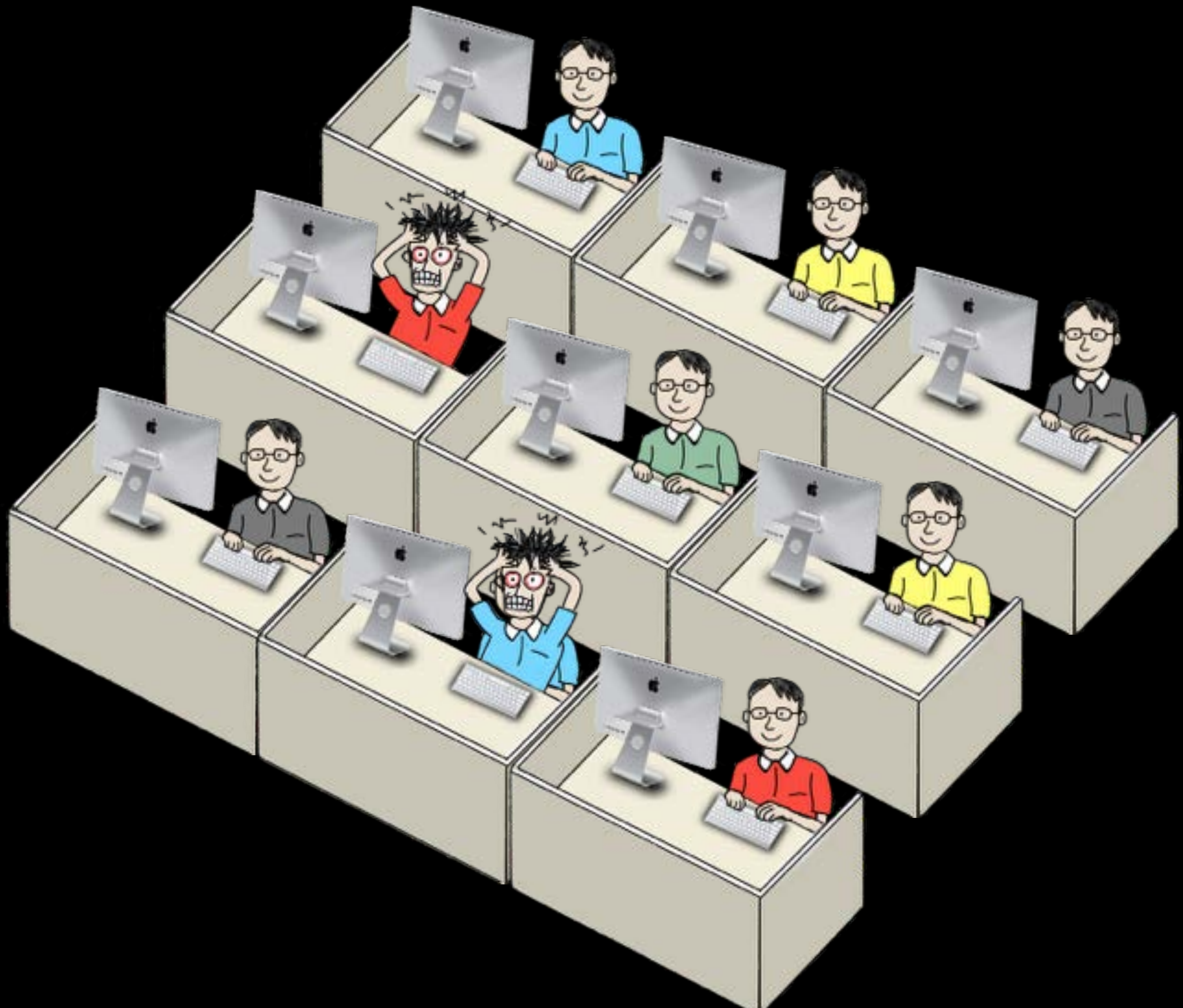
Software Design Process



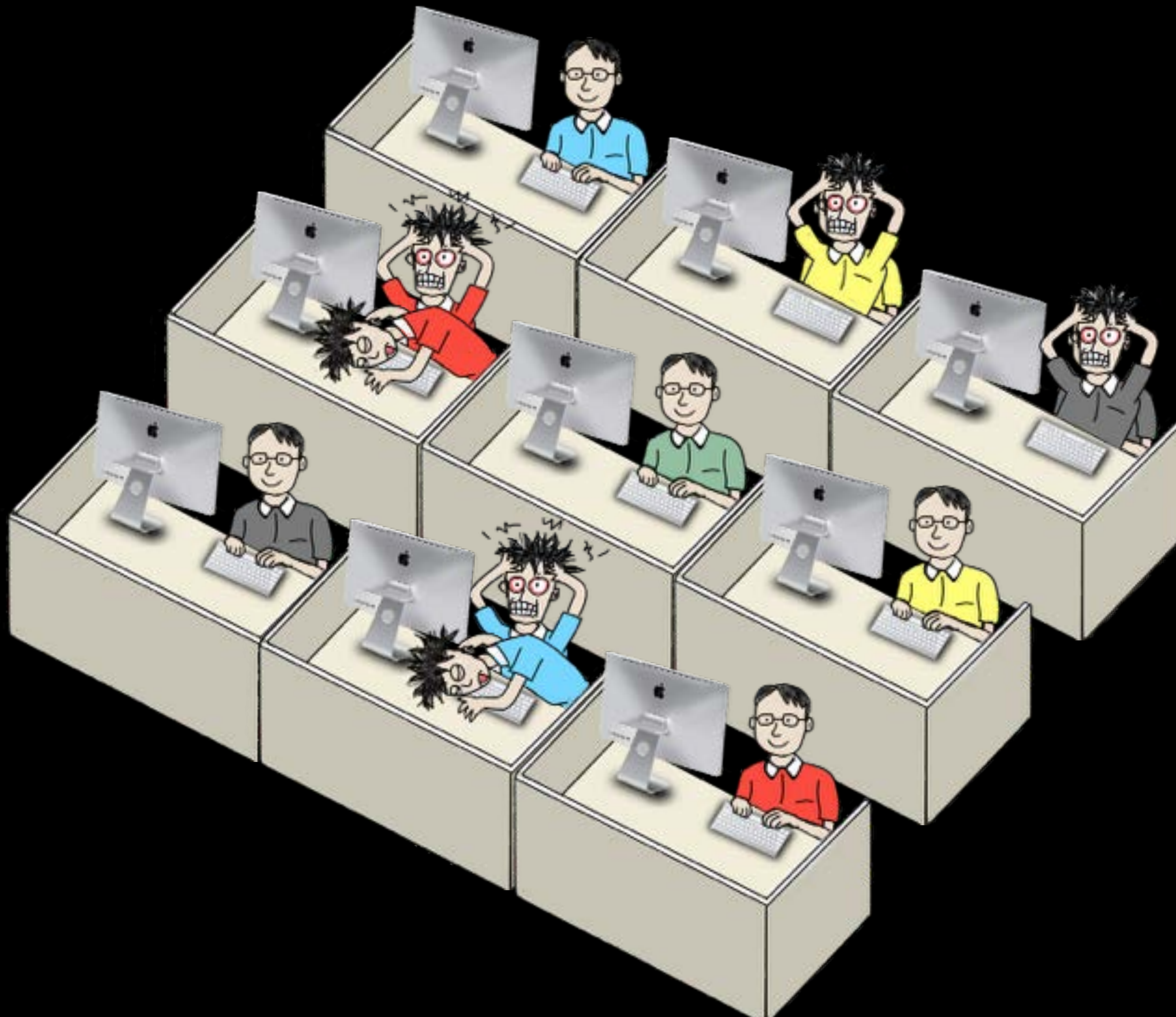
Software Design Process



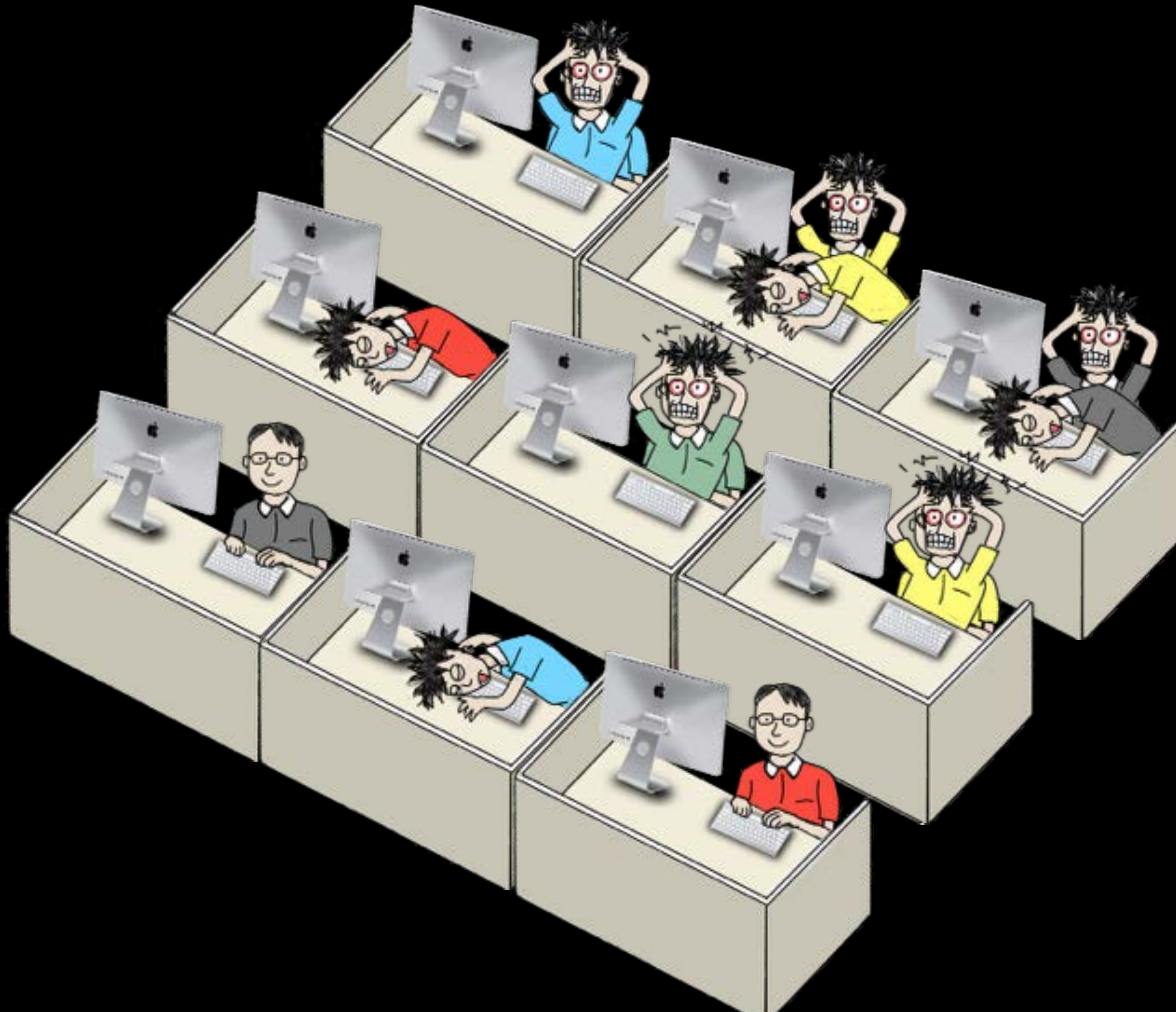
Software Design Process



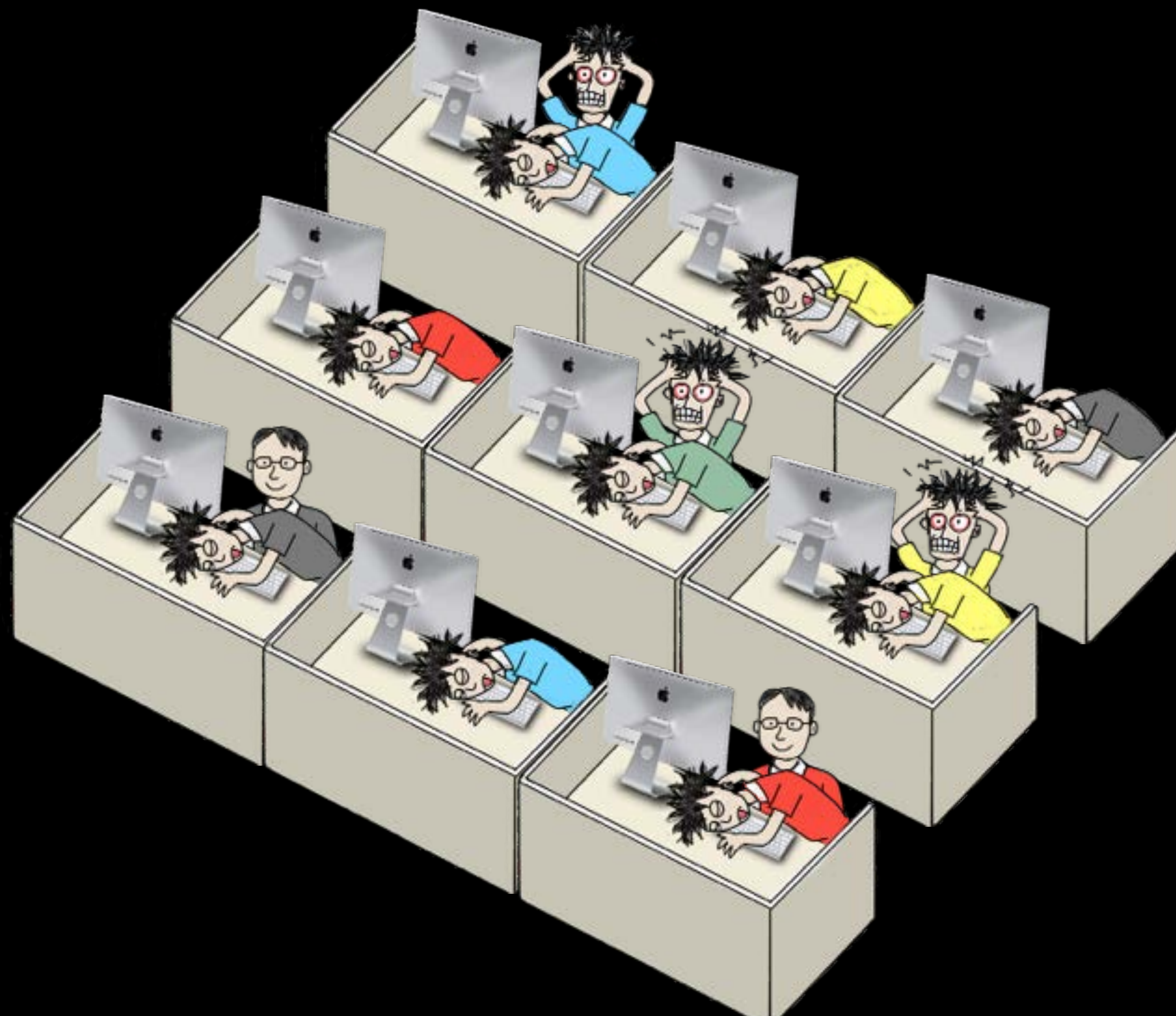
Software Design Process



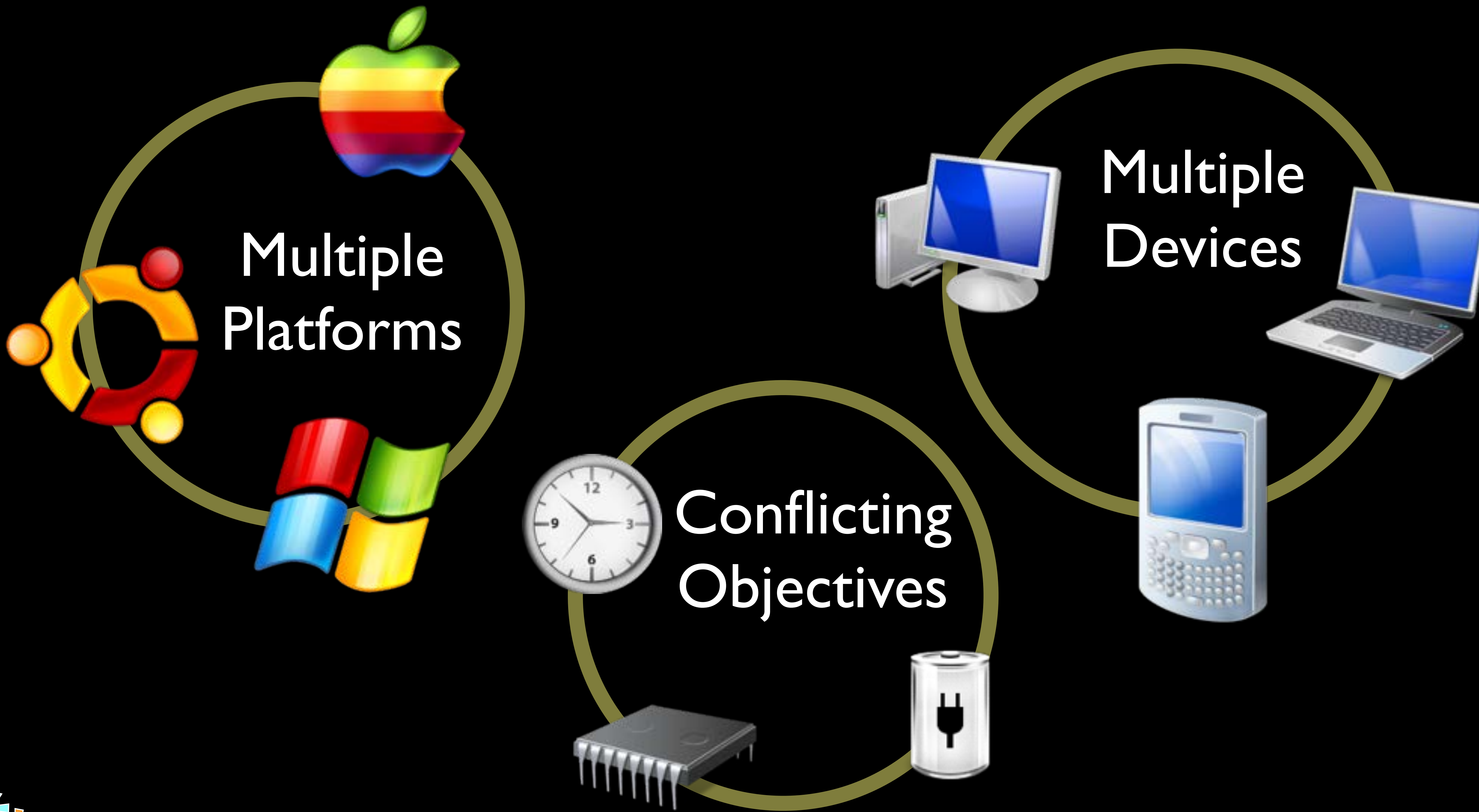
Software Design Process



Software Design Process



Multiplicity



Why is the programmer human?

Which requirements must be human coded ?

Functional Requirements

humans have to define these

Non-Functional Requirements

a machine can optimise these

Which requirements are essential to human ?

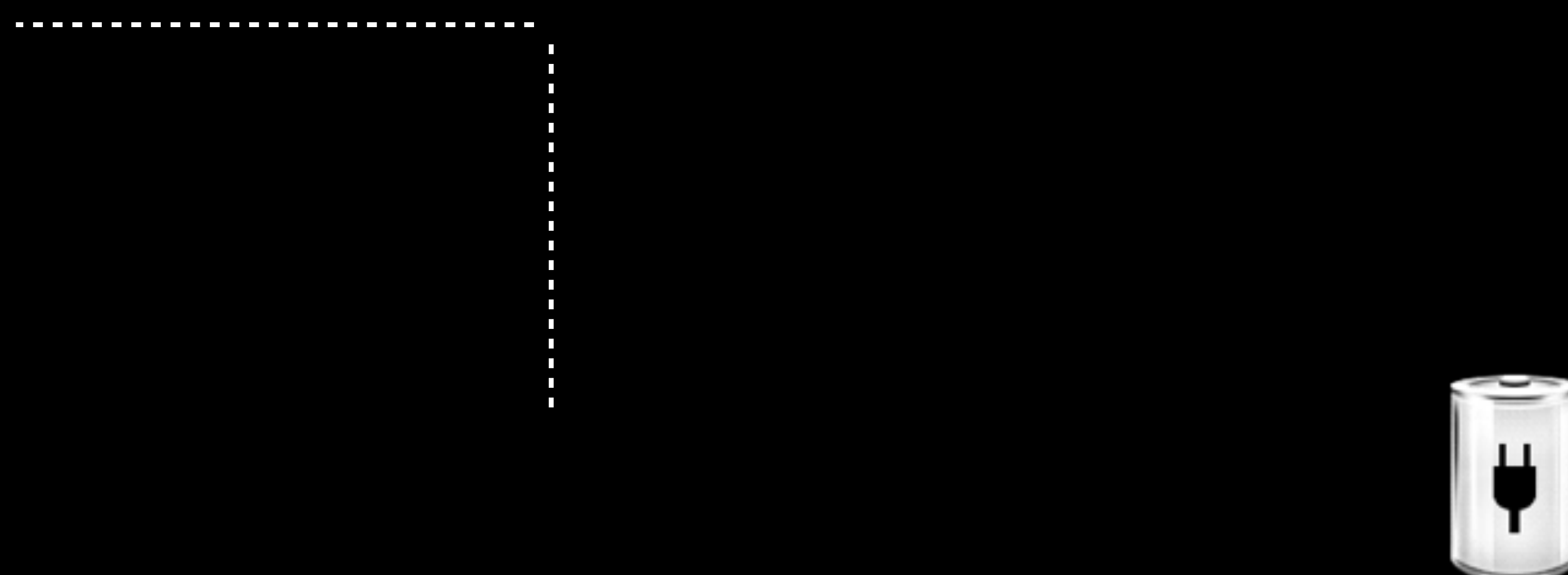
Functional Requirements

humans have to define these

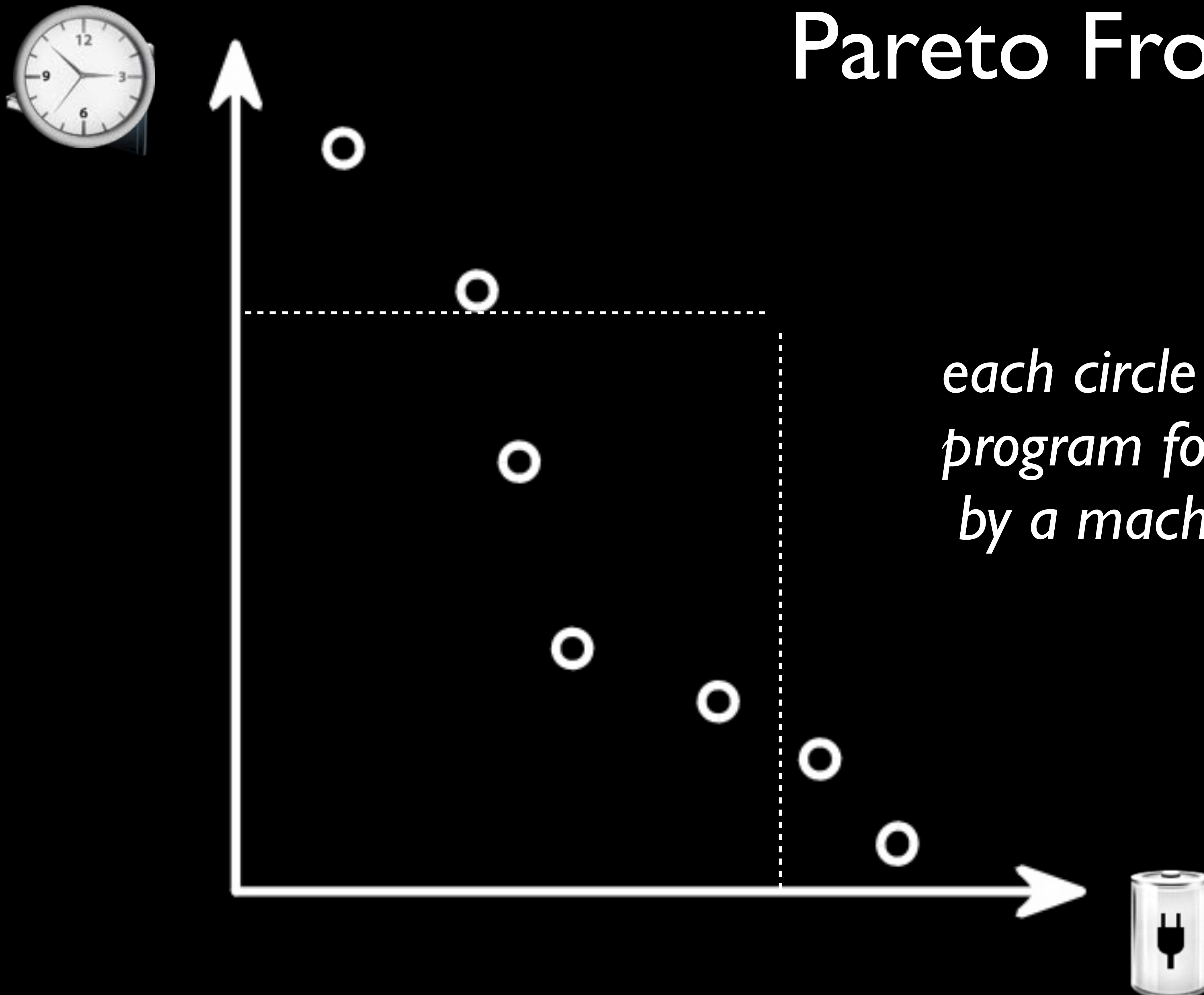
Non-Functional Requirements

a machine can optimise these

Pareto Front

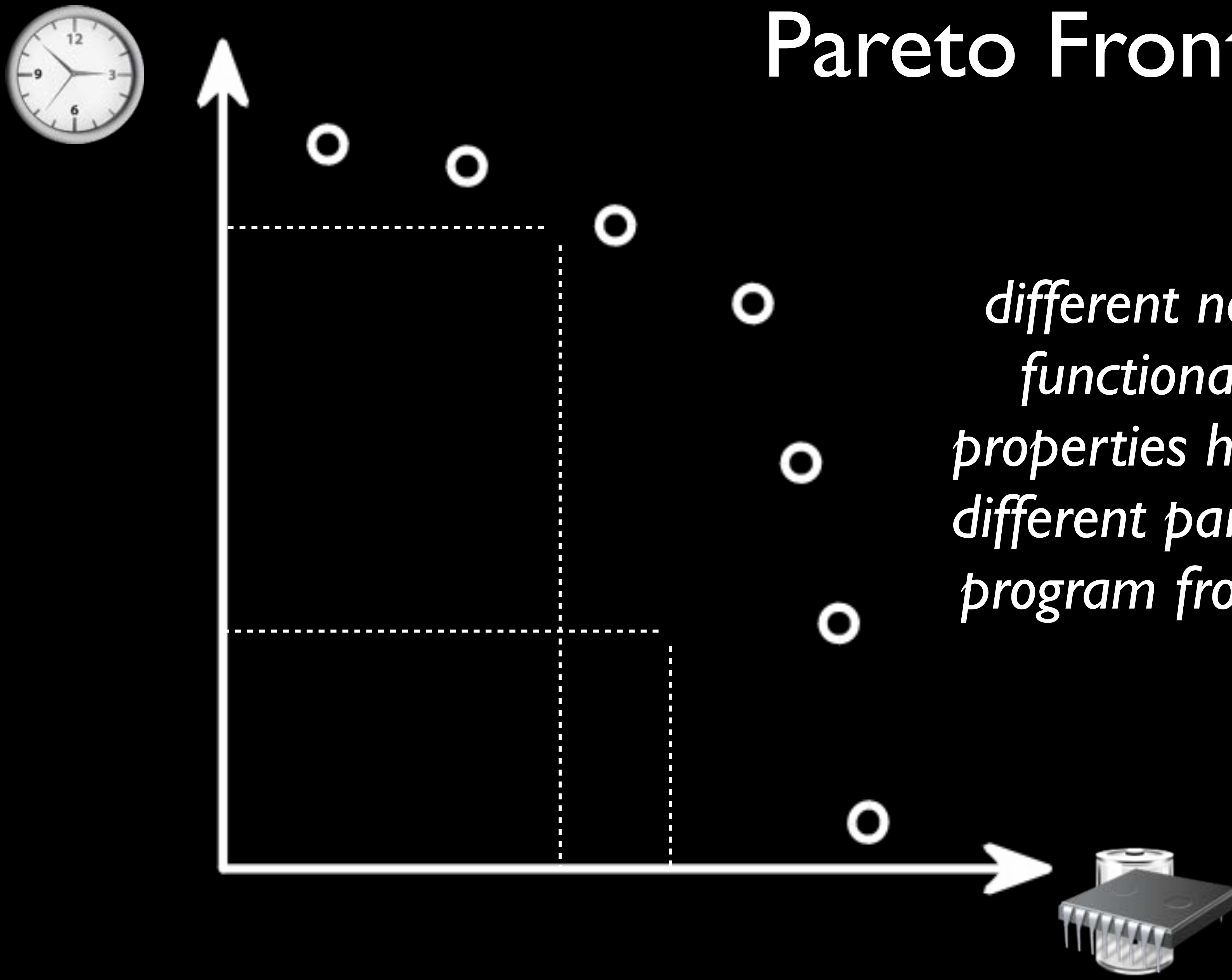


Pareto Front

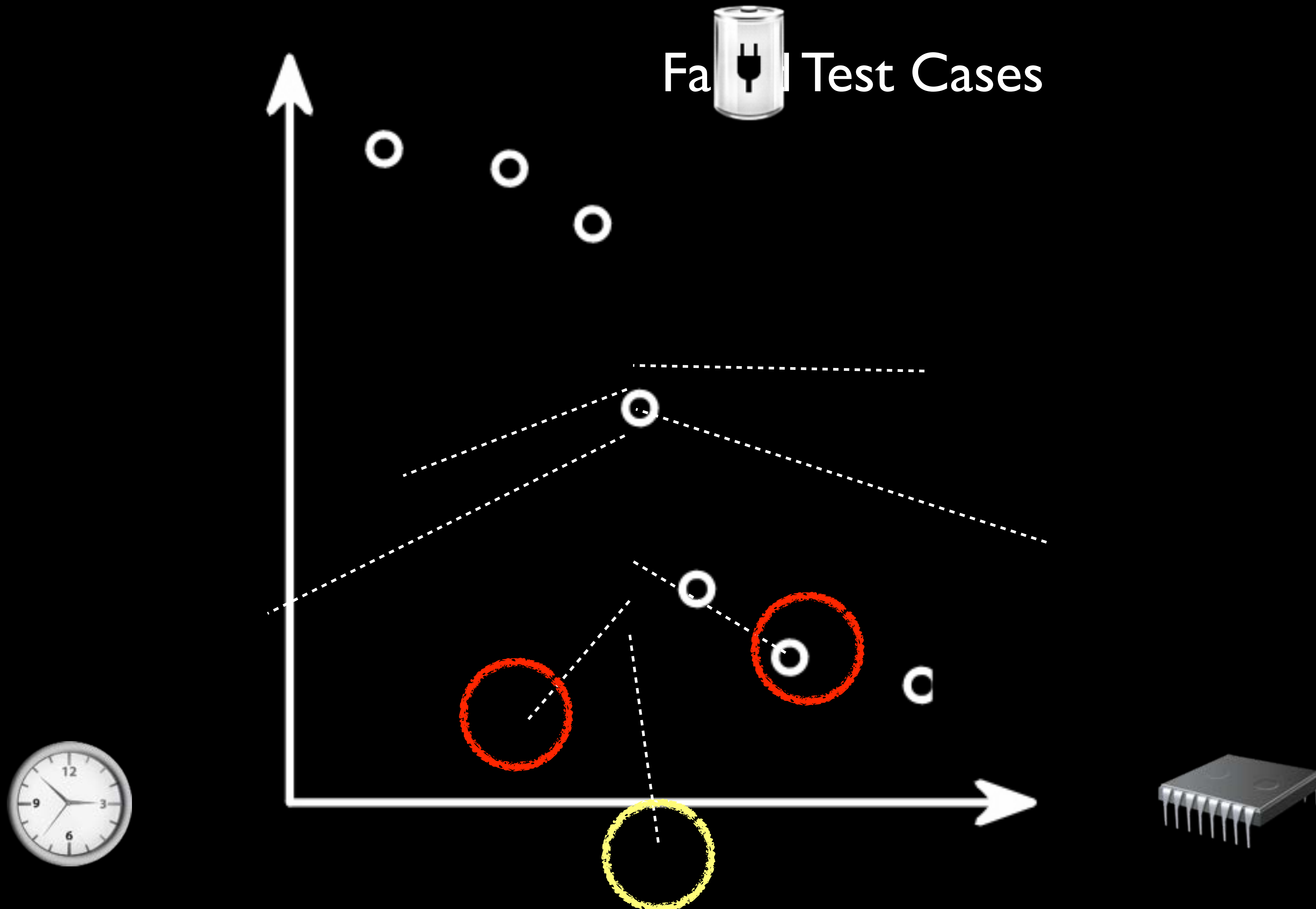


each circle is a program found by a machine

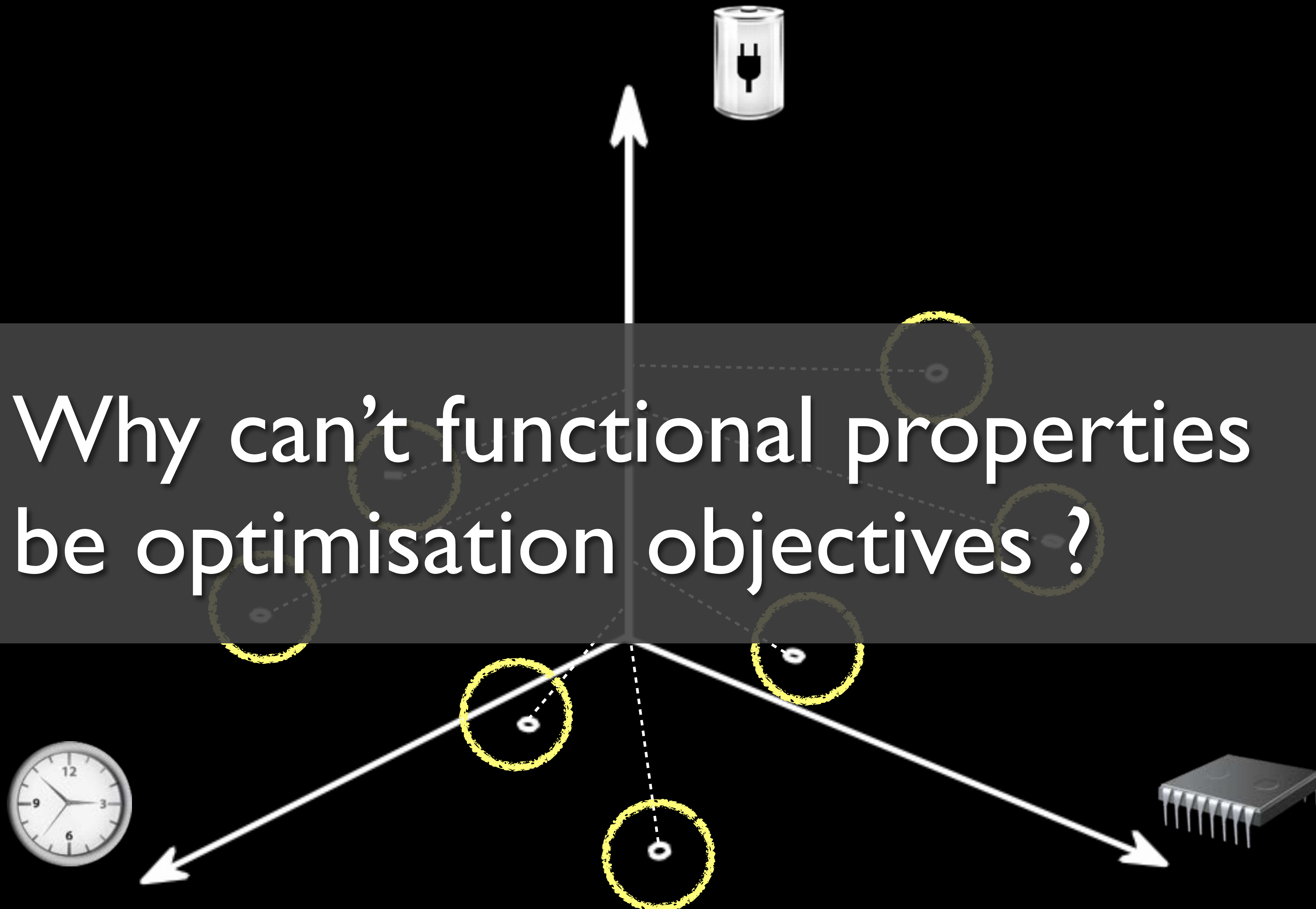
Pareto Front



different non functional properties have different pareto program fronts

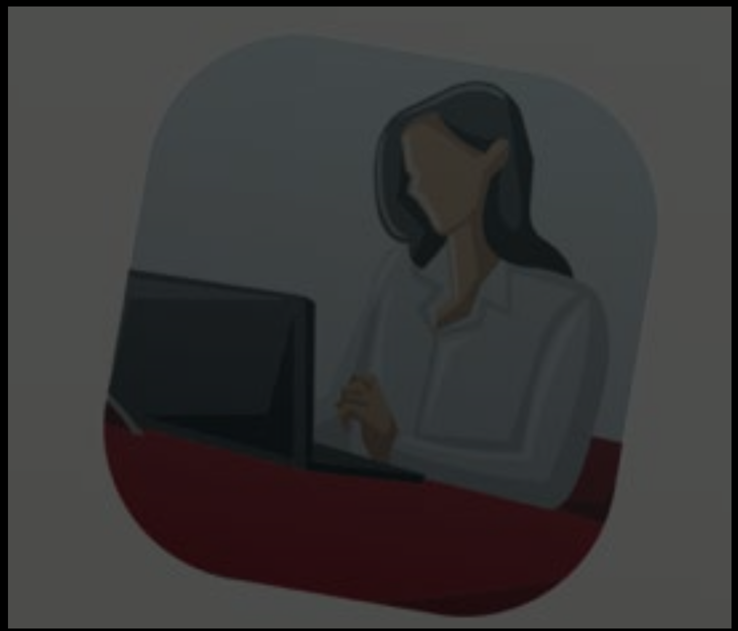

Fa l Test Cases

Why can’t functional properties be optimisation objectives ?

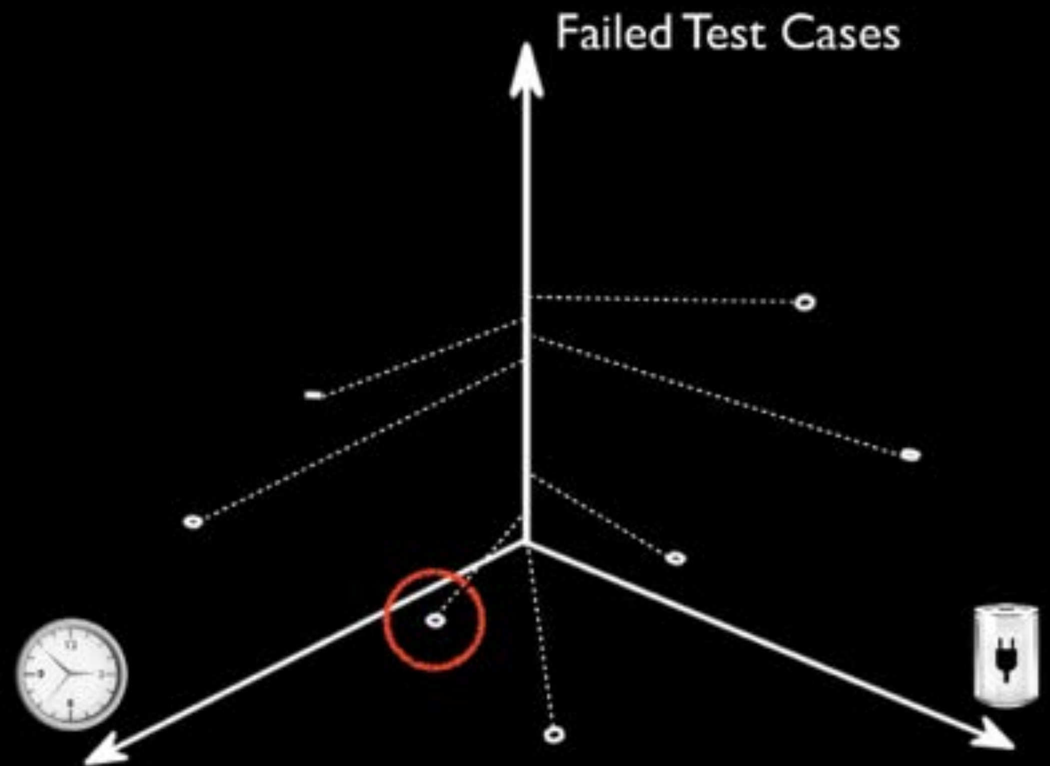


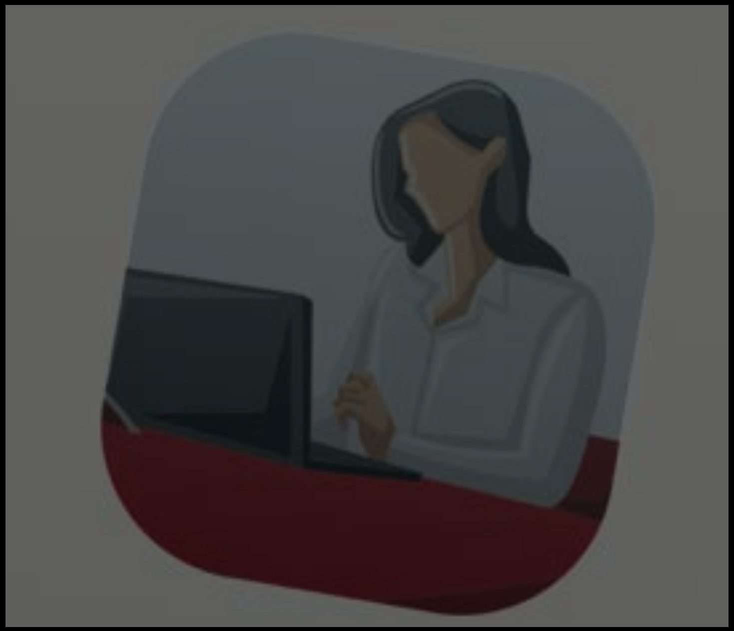
Optimisation

Optimisation

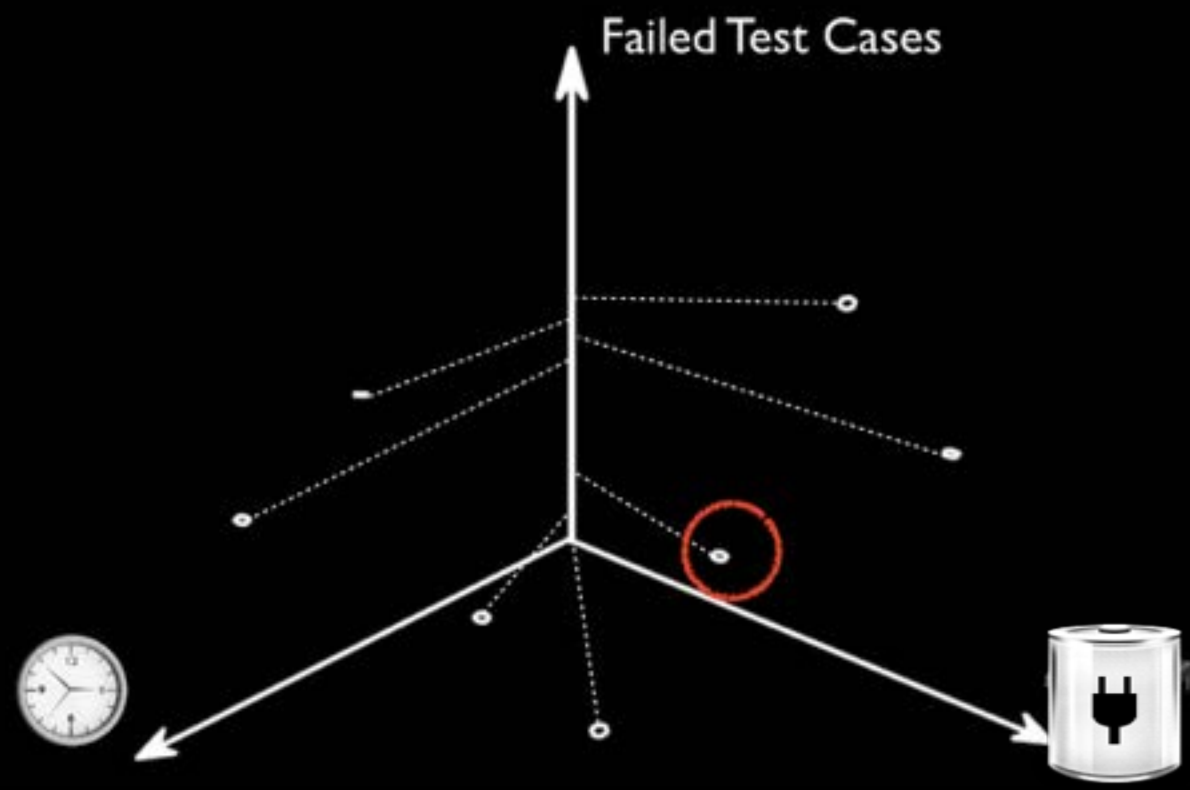


2.5 times faster but
failed 1 test case?



Optimisation



double the battery life
but failed 2 test cases?

Genetic Programming for Software Transplantation

Genetic Programming for Software Transplantation

covered in more detail in the
WCRE 2013
keynote paper

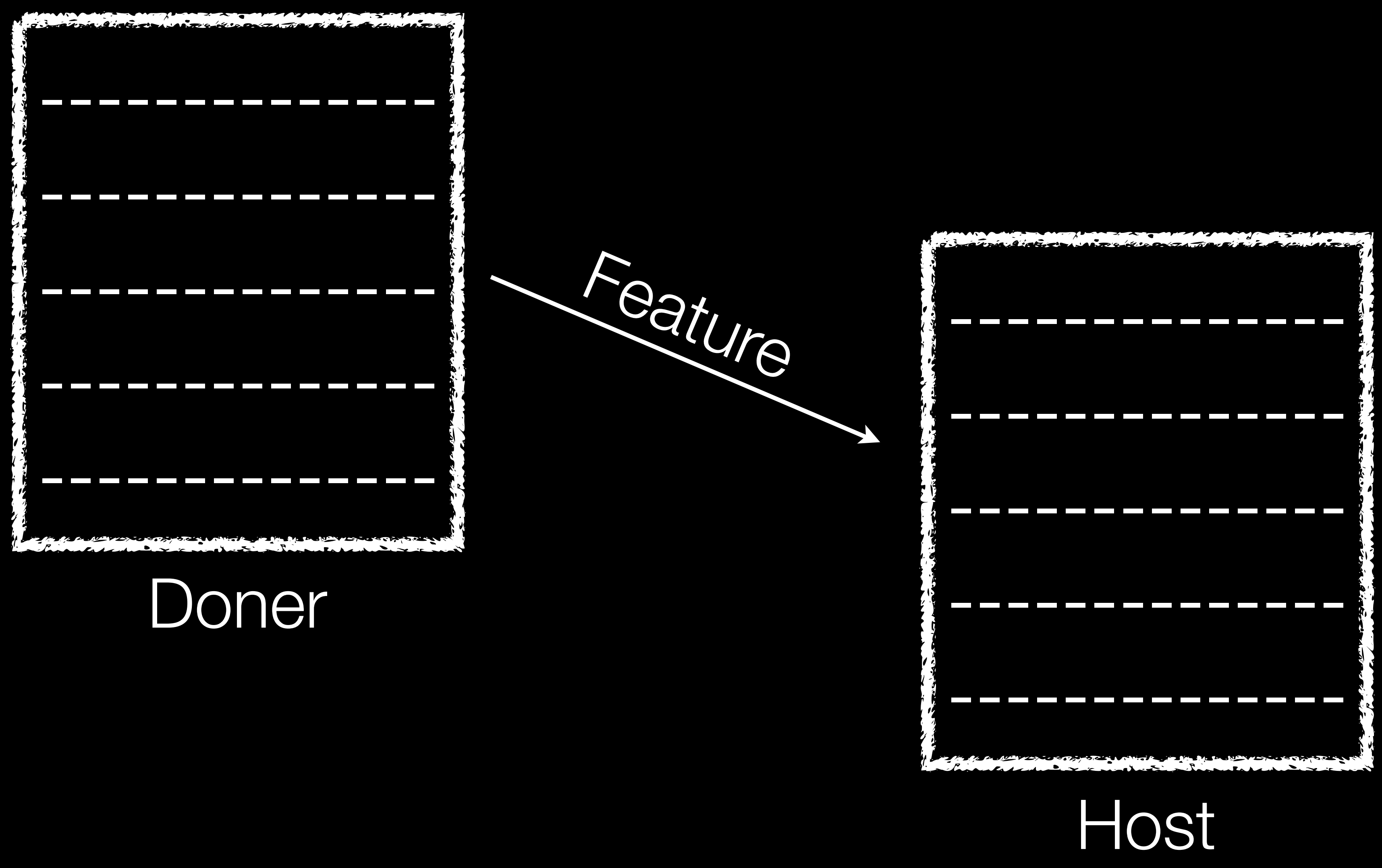
GP for Transplants

GP for Transplants

L A T I I V R

GP for Transplants

L
A
T
I
I
V
R



GP for Transplants

Localise

A

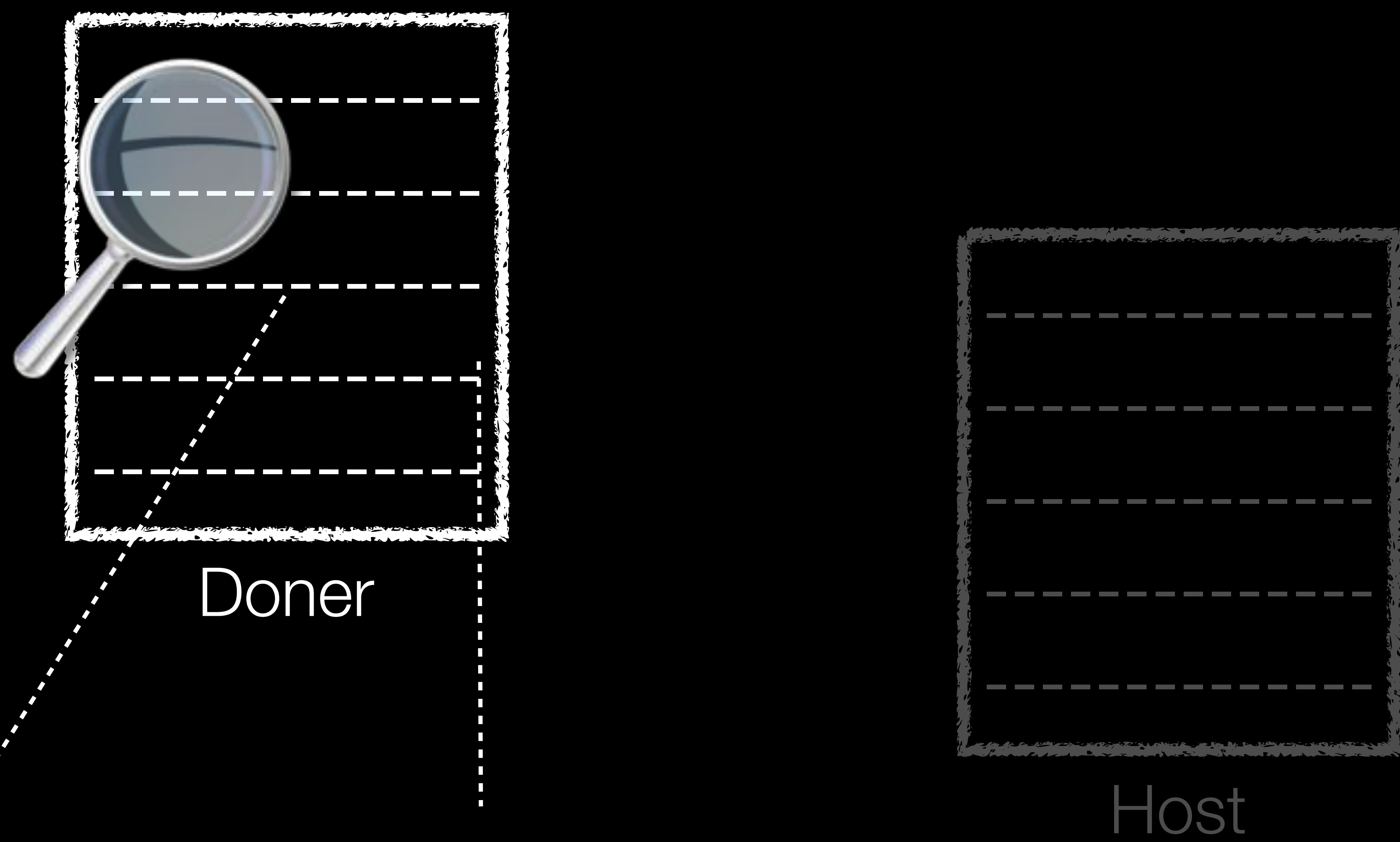
T

I

I

V

R




```
if(ptr!=NULL)
  foo(*ptr);
```

GP for Transplants

Localise

Abstract

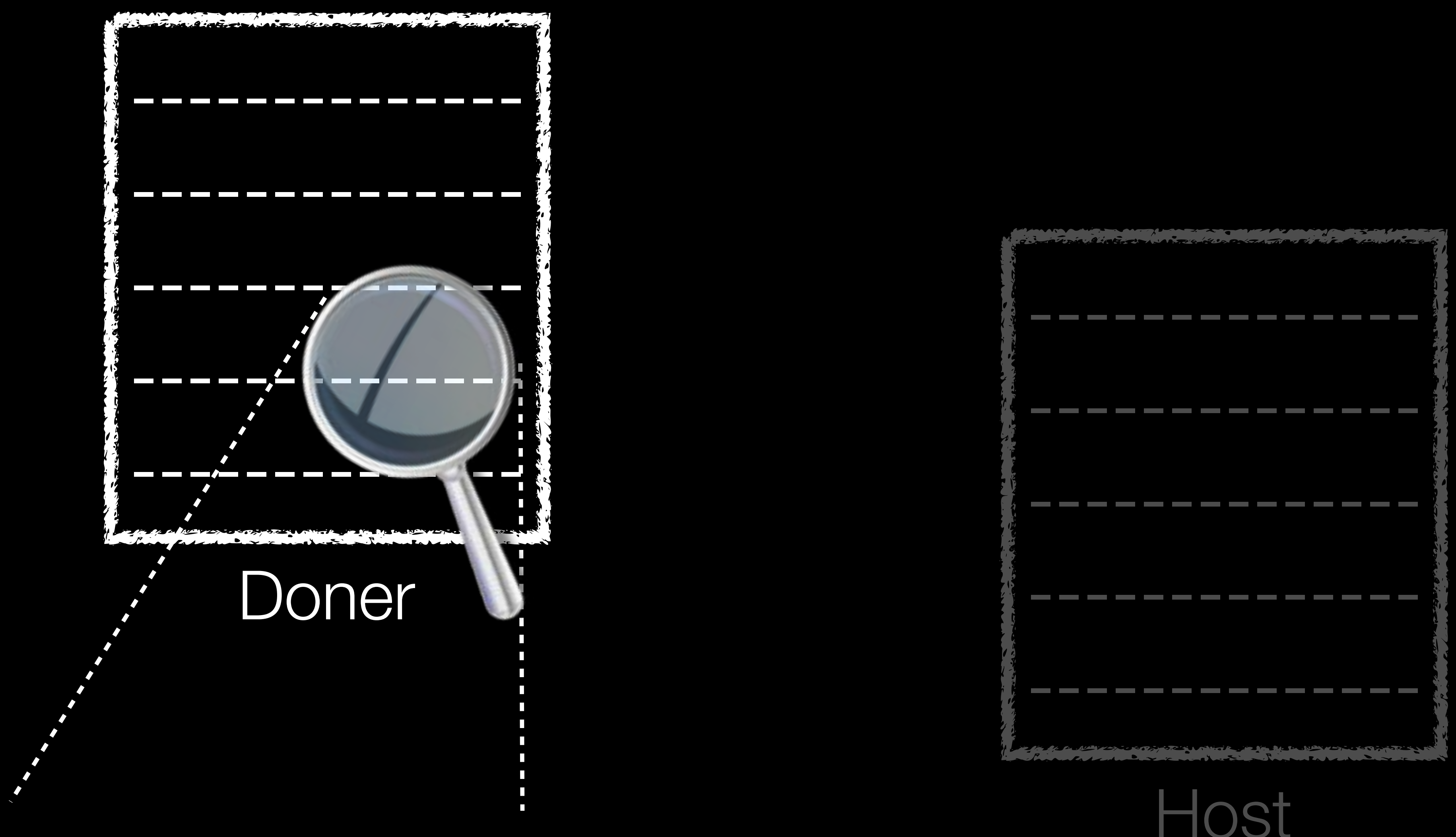
T

I

I

V

R




```
if(ptr!=NULL)
  foo(*ptr);
```

→

```
if( □1 !=NULL)
  □2 ( □1 );
```

GP for Transplants

Localise

Abstract

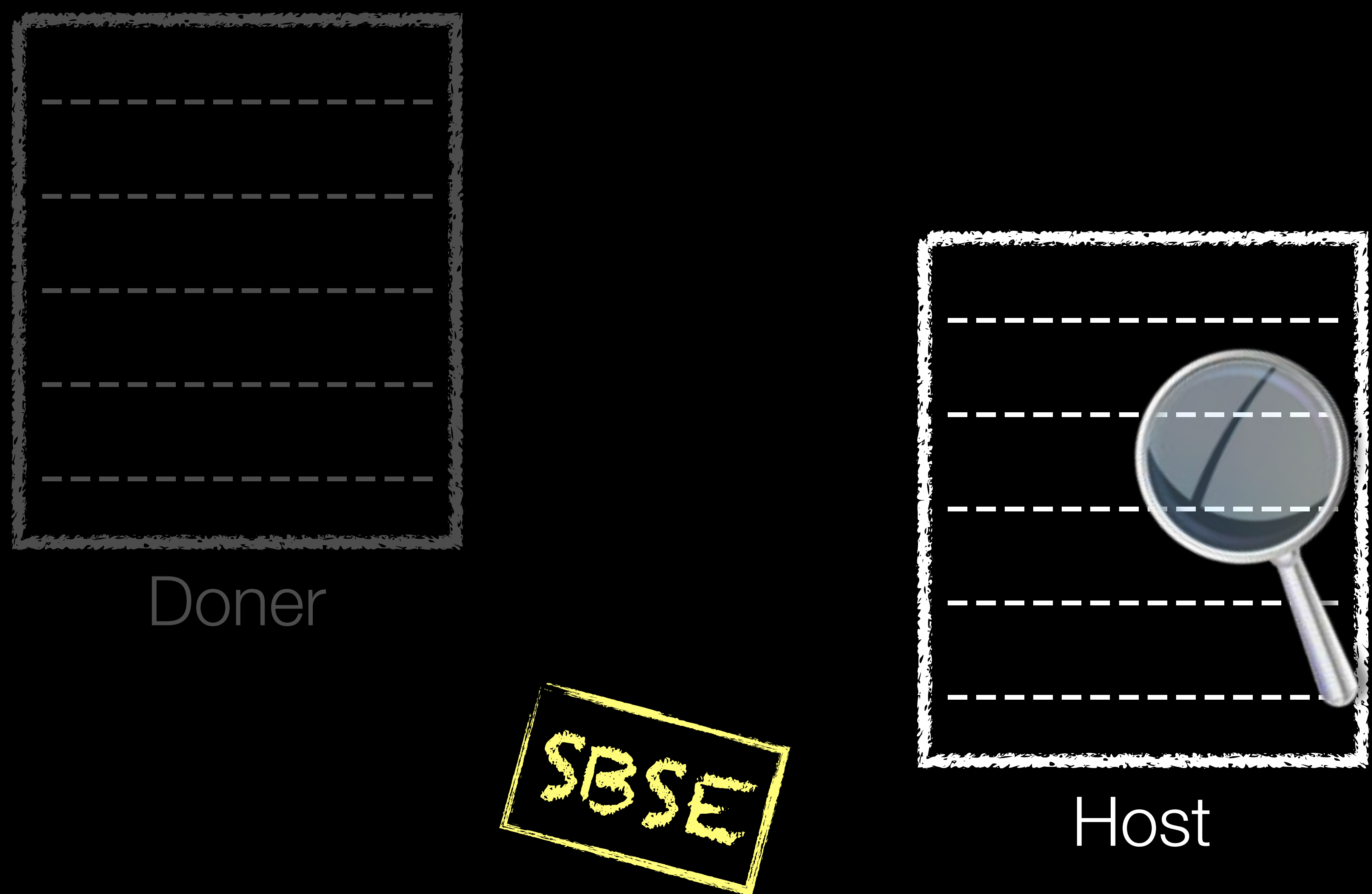
Target

I

I

V

R



GP for Transplants

Localise

Abstract

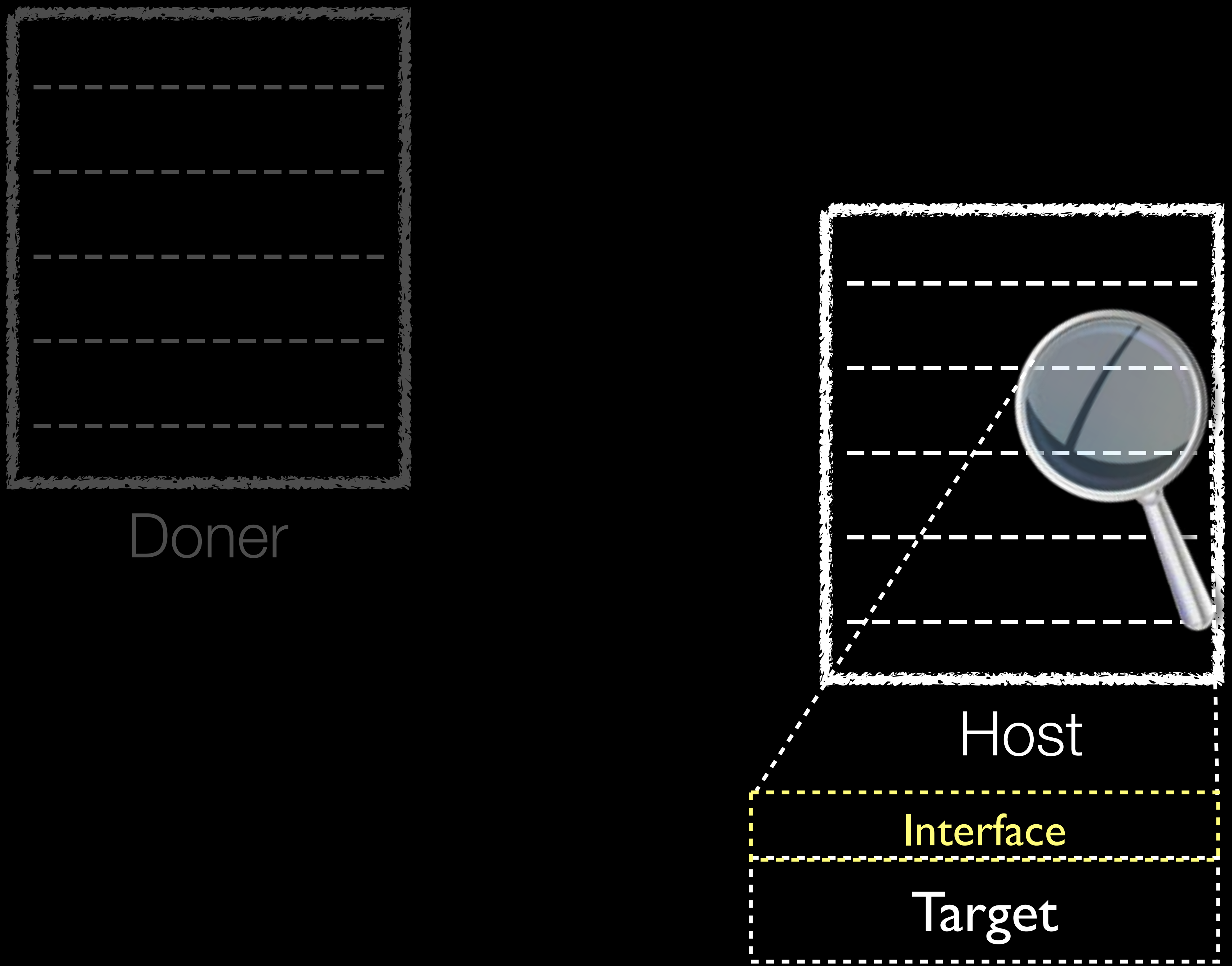
Target

Interface

I

V

R



GP for Transplants

Localise

Abstract

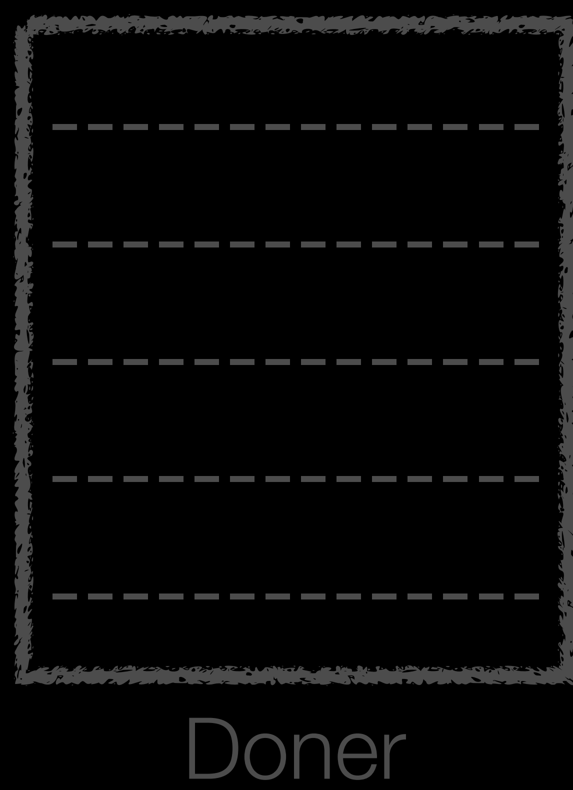
Target

Interface

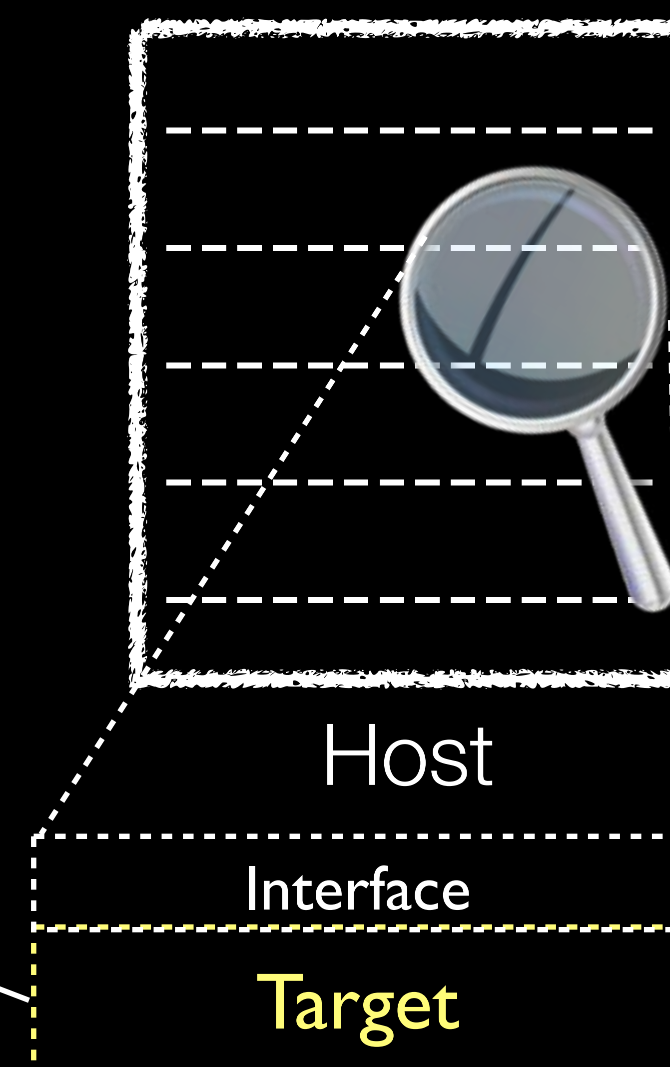
Insert

V

R




```
if(□1 !=NULL)
  □2 ( □1 );
```



GP for Transplants

Localise

Abstract

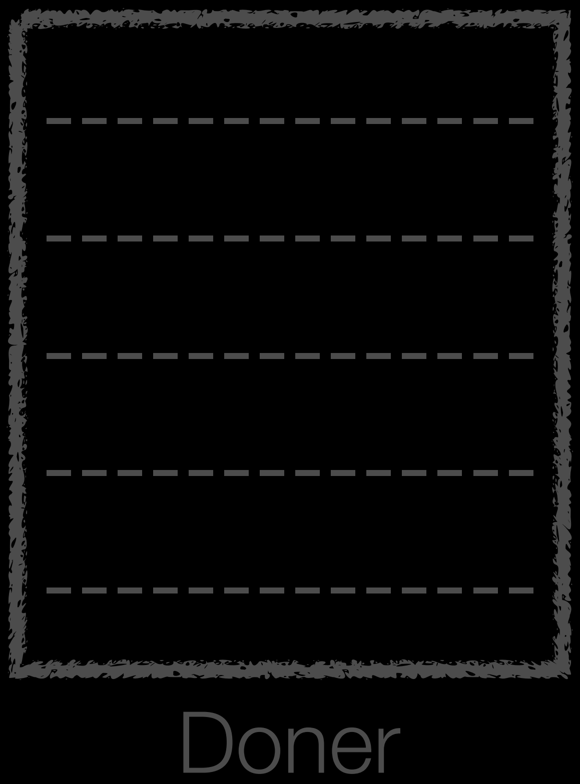
Target

Interface

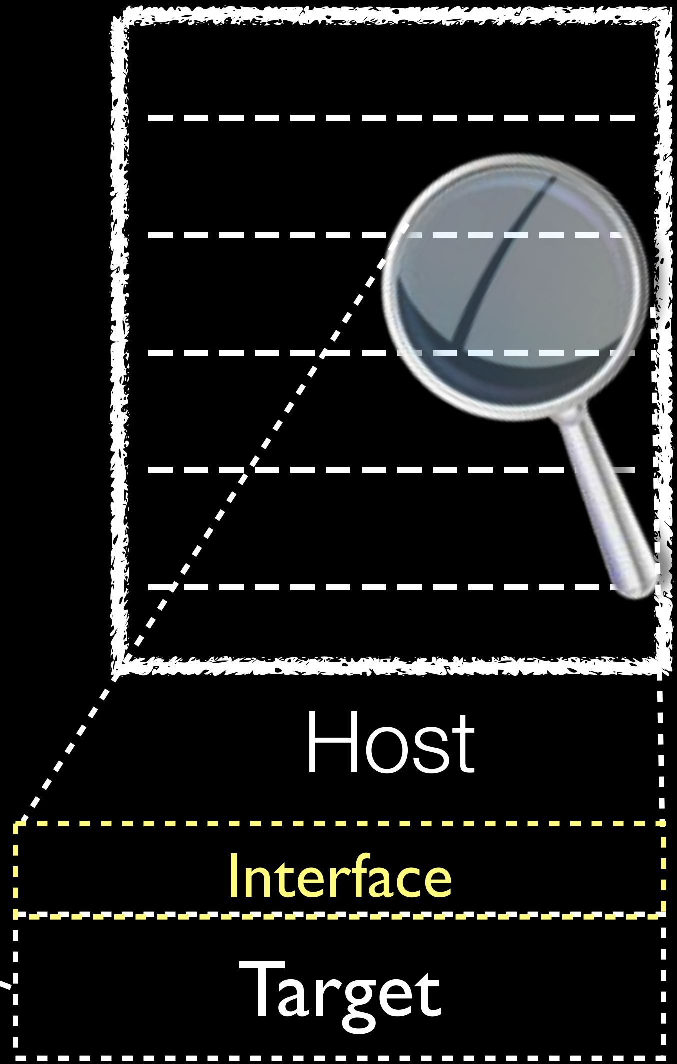
Insert

V

R




```
if( host_ptr!=NULL)
host_action(*host_ptr);
```



GP for Transplants

Localise

Abstract

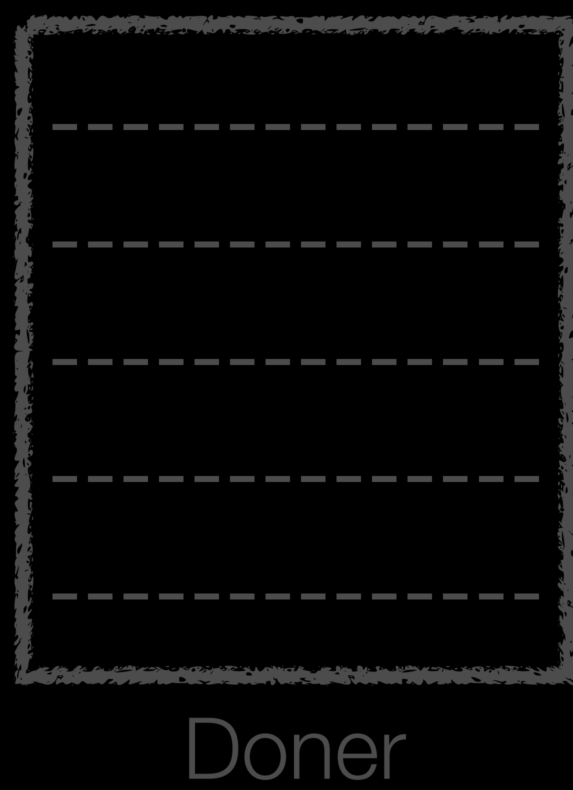
Target

Interface

Insert

Validate

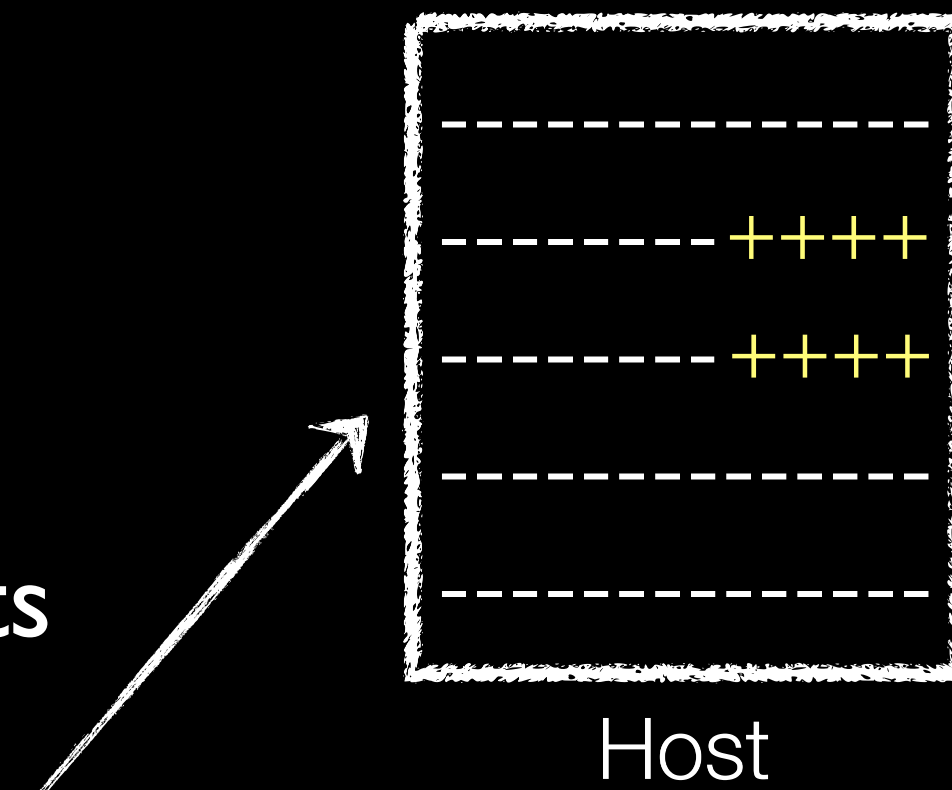
R



1 new feature tests

2 regression tests

3 quality tests



GP for Transplants

Localise

Abstract

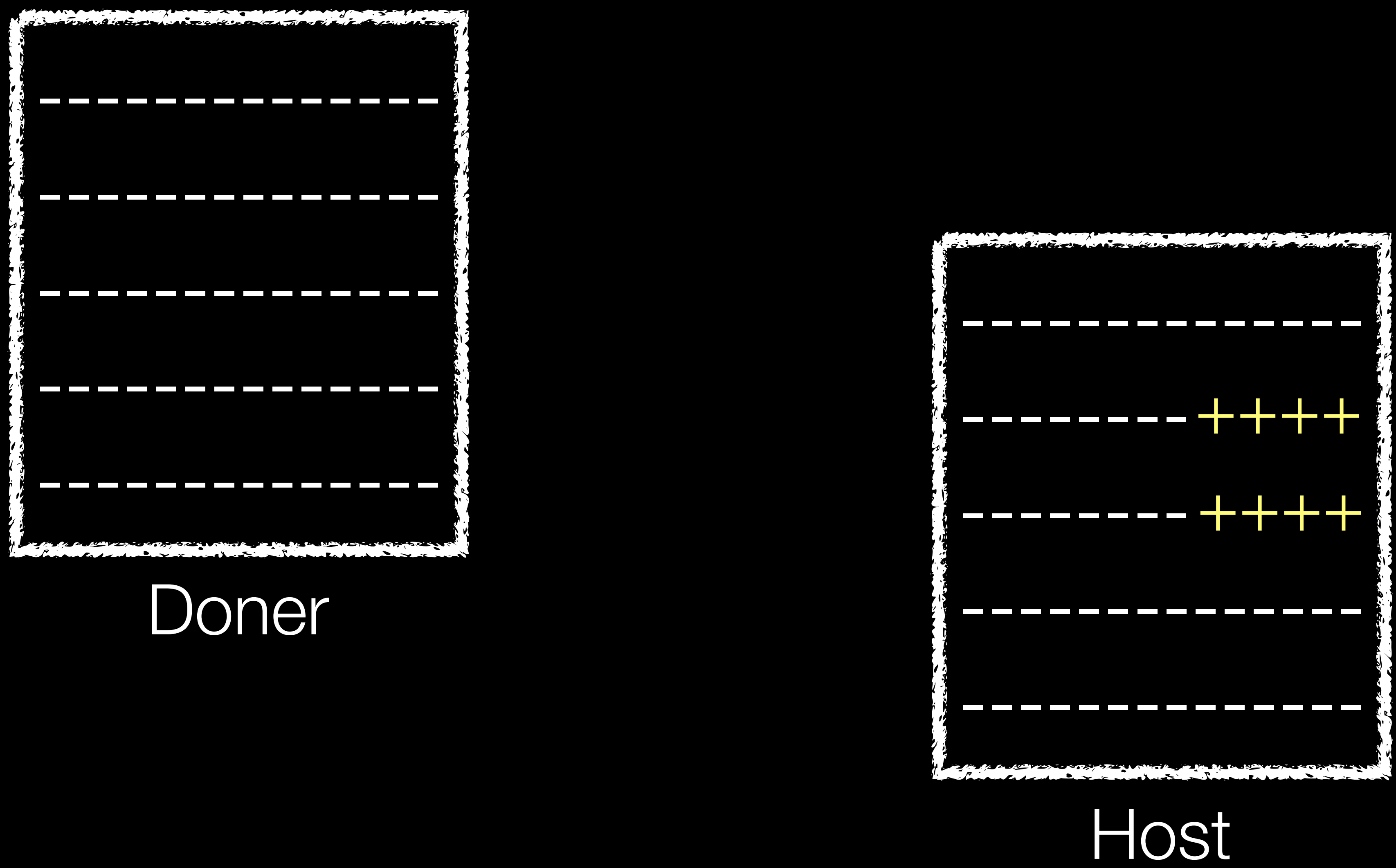
Target

Interface

Insert

Validate

Repeat

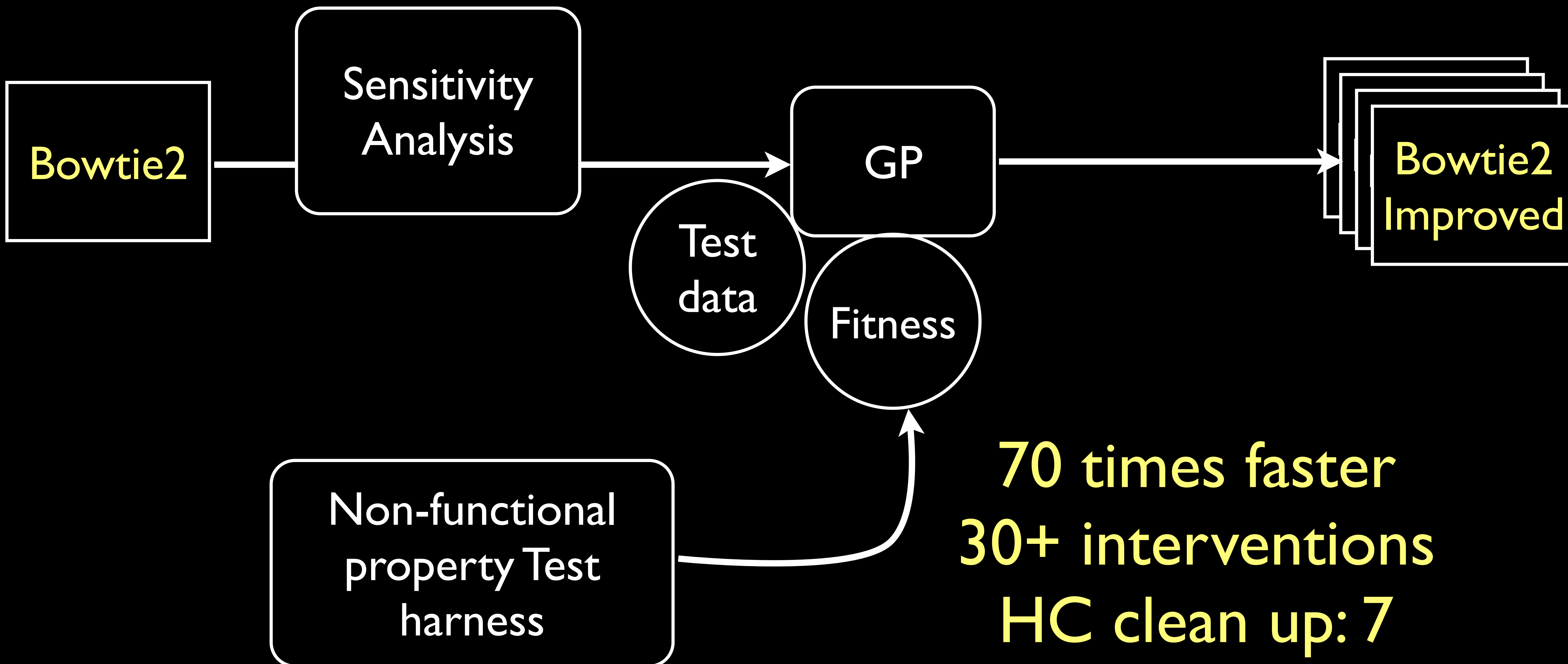


Offline Genetic Improvement

Offline Genetic Improvement

… what have we managed to achieve so far …

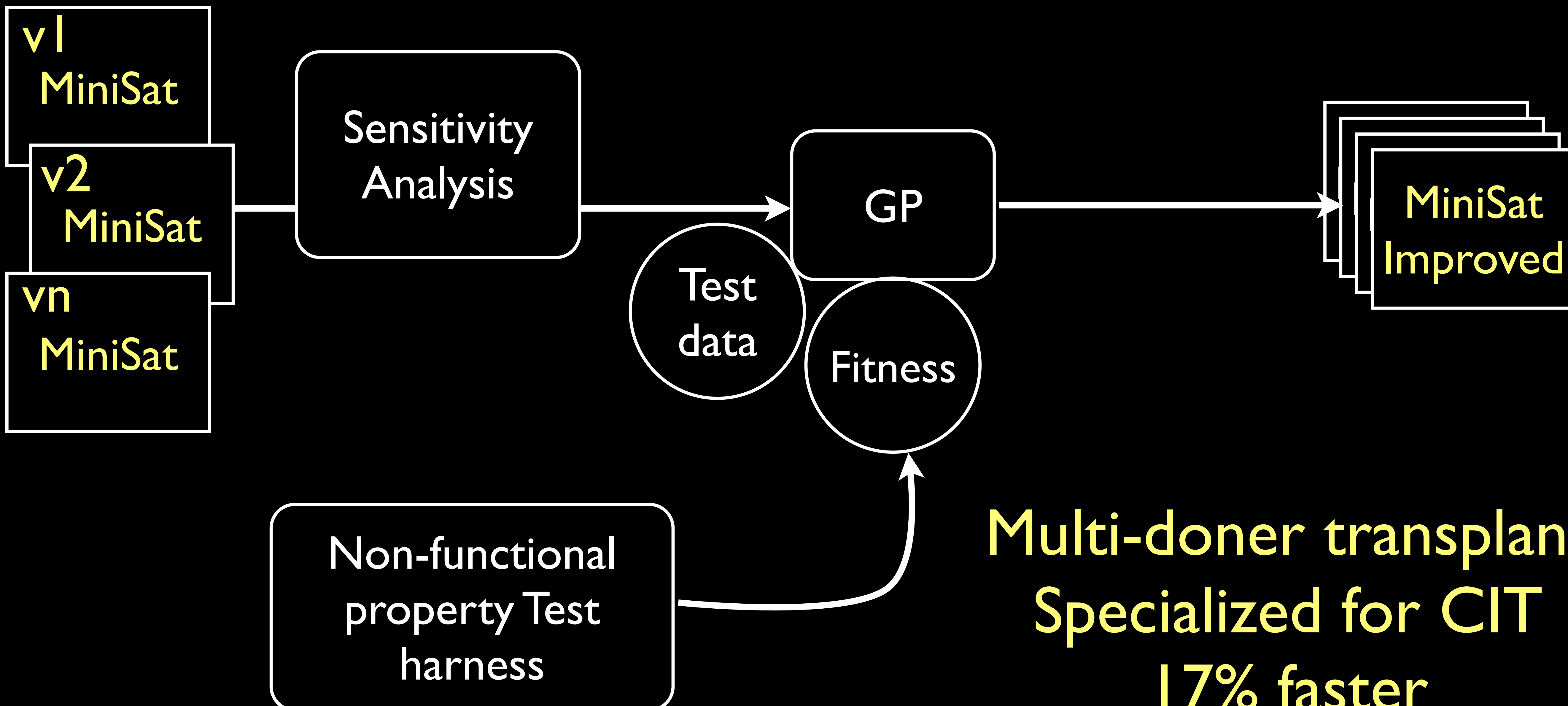
Genetic Improvement of Programs



70 times faster
30+ interventions
HC clean up: 7
slight semantic improvement

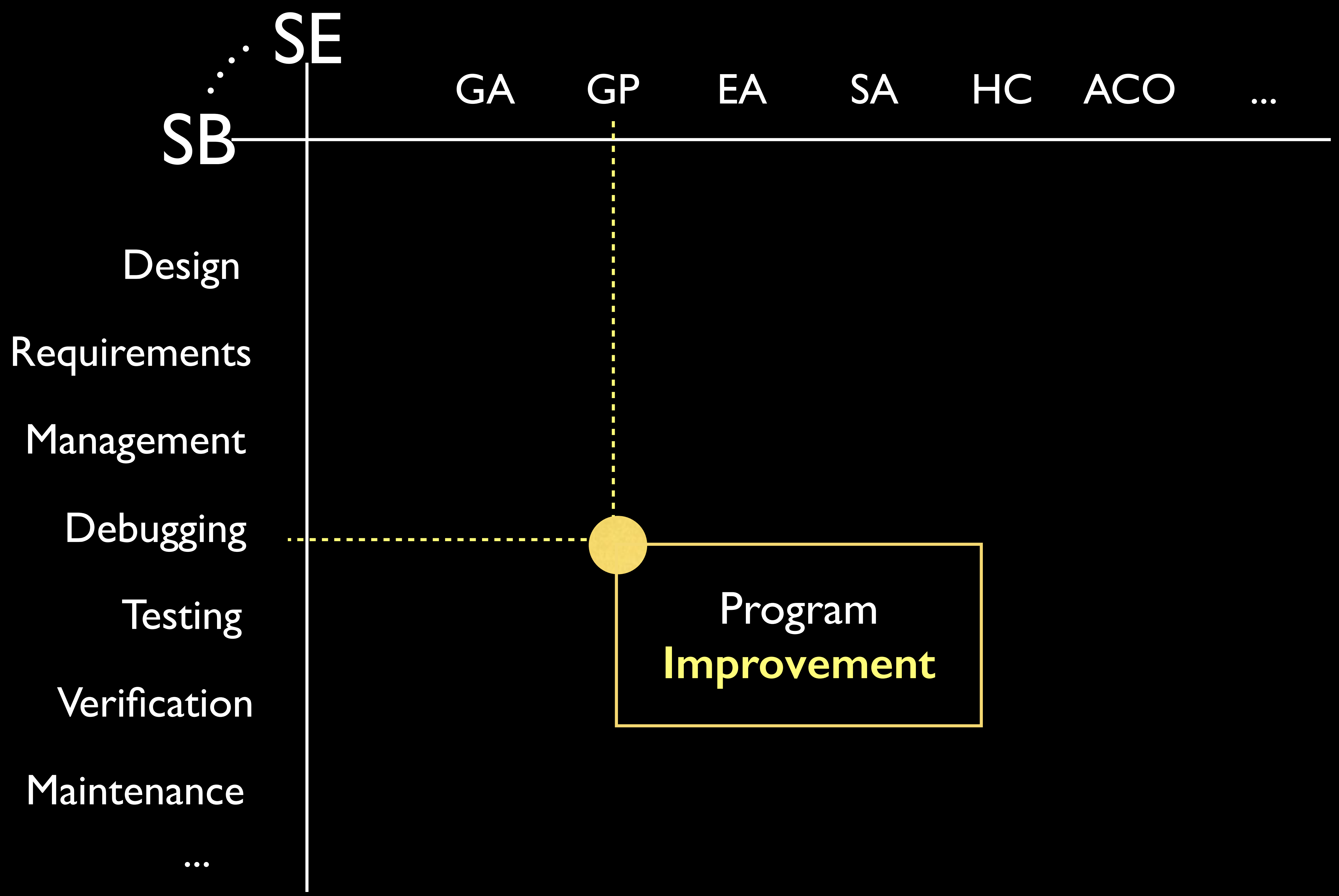
W. B. Langdon and M. Harman
Optimising Existing Software with Genetic Programming. TEC 2014 (TR available)

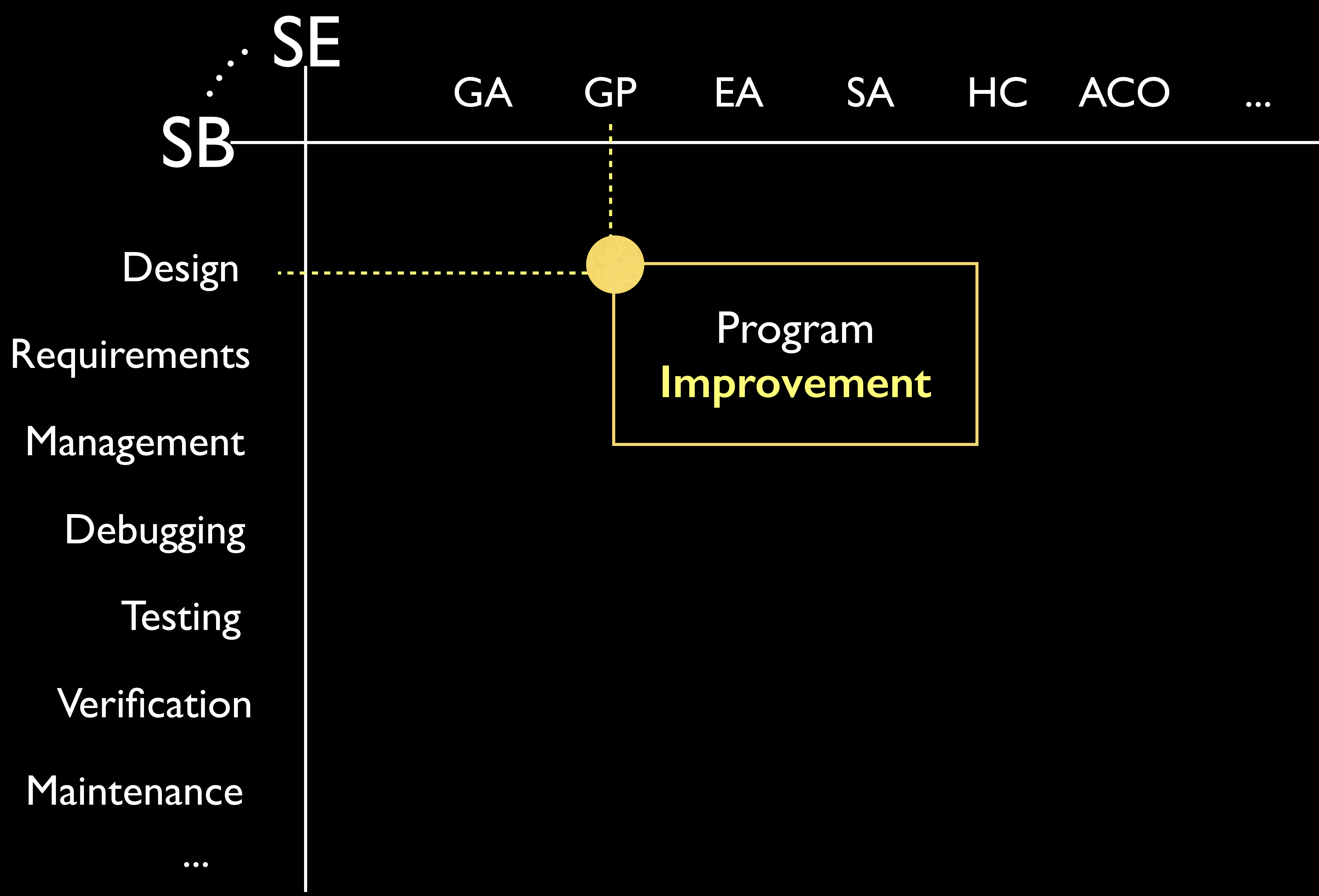
Genetic Improvement of Programs



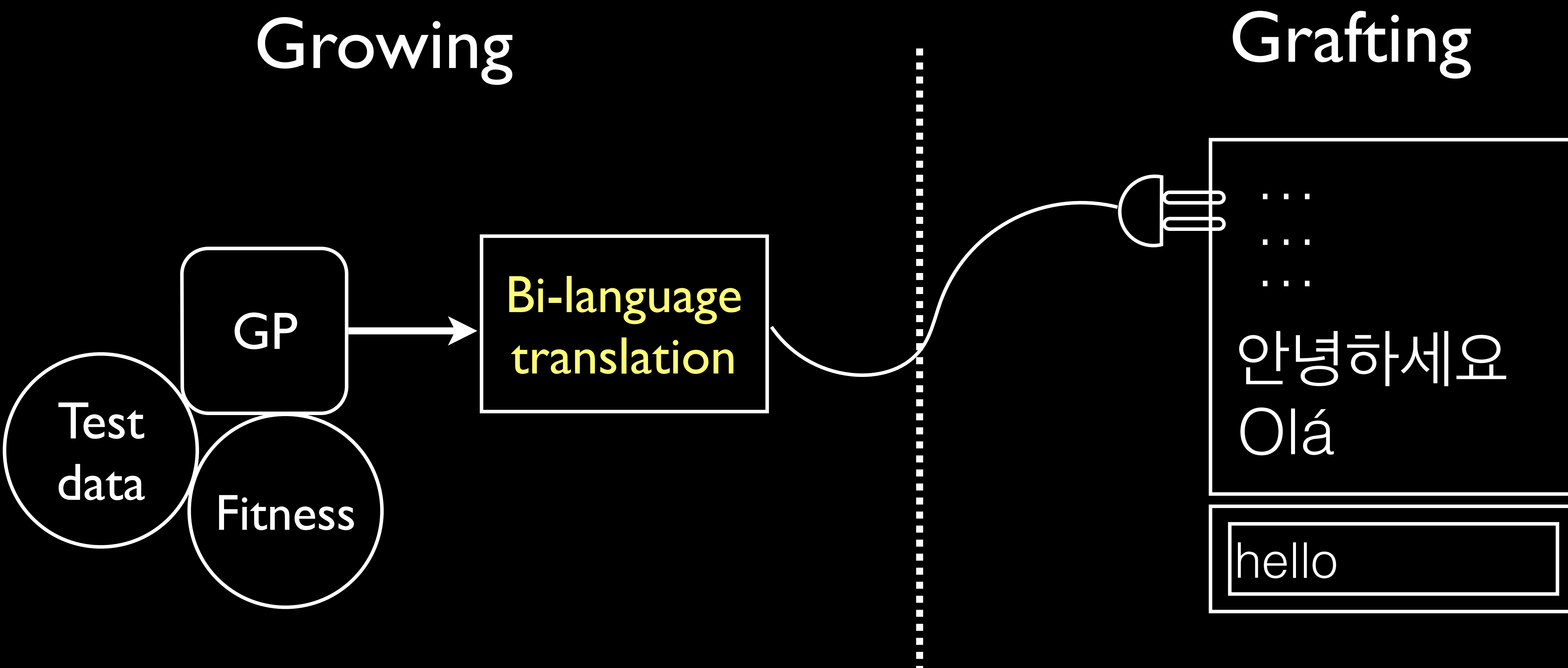
Multi-doner transplant
Specialized for CIT
17% faster

Justyna Petke, Mark Harman, William B. Langdon and Westley Weimer
Using Genetic Improvement & Code Transplants to Specialise a C++ program to a Problem Class (EuroGP'14)


SE
SB
GA
GP
EA
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Program
Improvement


SE
SB
GA
GP
EA
SA
HC
ACO
...
Design
Requirements
Management
Debugging
Testing
Verification
Maintenance
...
Program
Improvement

Genetic Improvement
Babel Pidgin: Grow new functionality



Mark Harman, Yue Jia and William B Langdon
Babel Pidgin: SBSE can grow and graft entirely new functionality into a real world system (SSBSE'14 Challenge)

Online Genetic Improvement

Online Genetic Improvement

This is described in the SEAMS 2014 keynote paper

Online phase

Offline phase

User specify operation characteristics

Online phase

Offline phase

User specify operation characteristics

Environmental and usage profile Learning

Phase 1

Data Collection

Online phase

Offline phase

User specify operation characteristics

Phase 1

Data Collection

Environmental and usage profile Learning

Tuneable Parameters

Program

Online phase

Offline phase

User specify operation characteristics

Phase 1

Data Collection

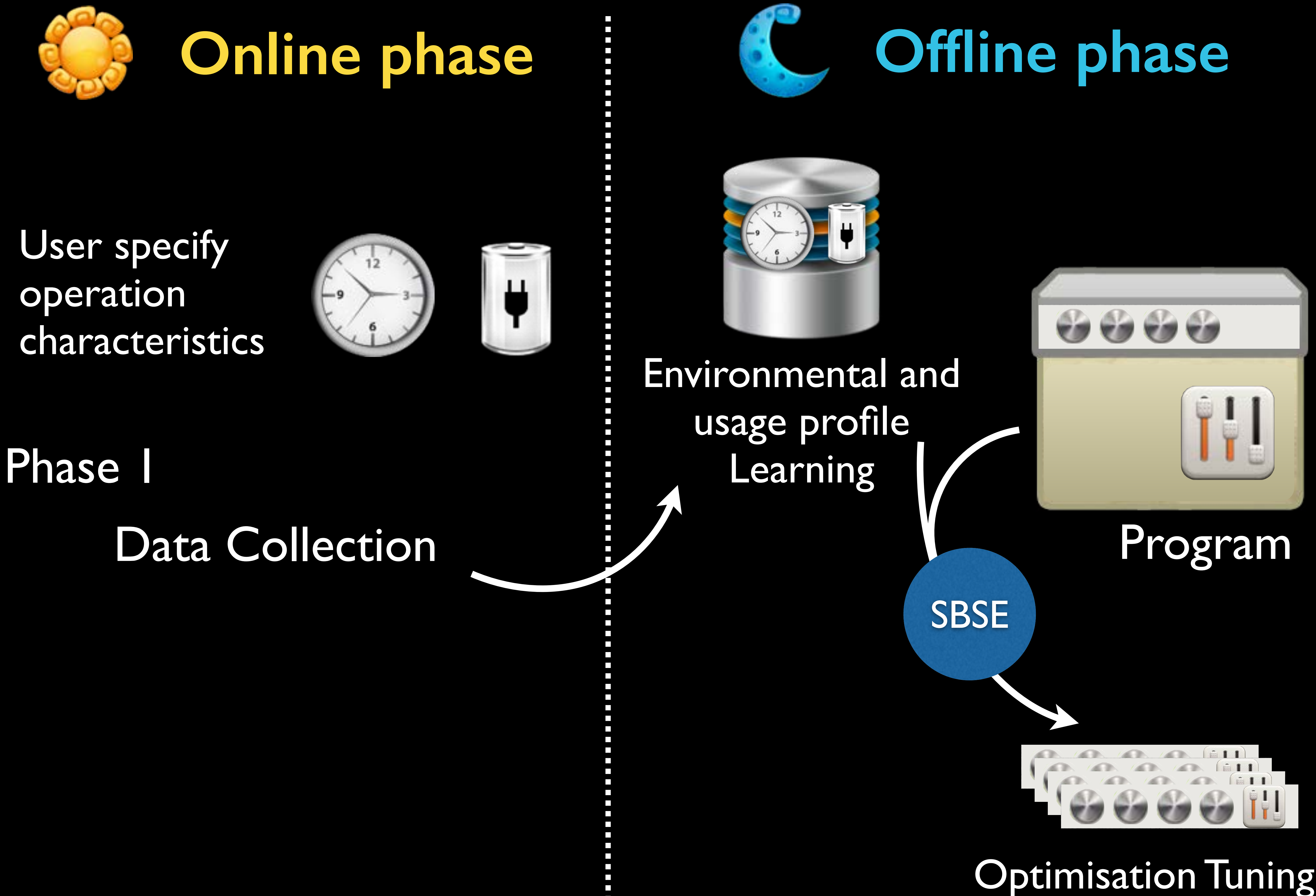
Environmental and usage profile Learning

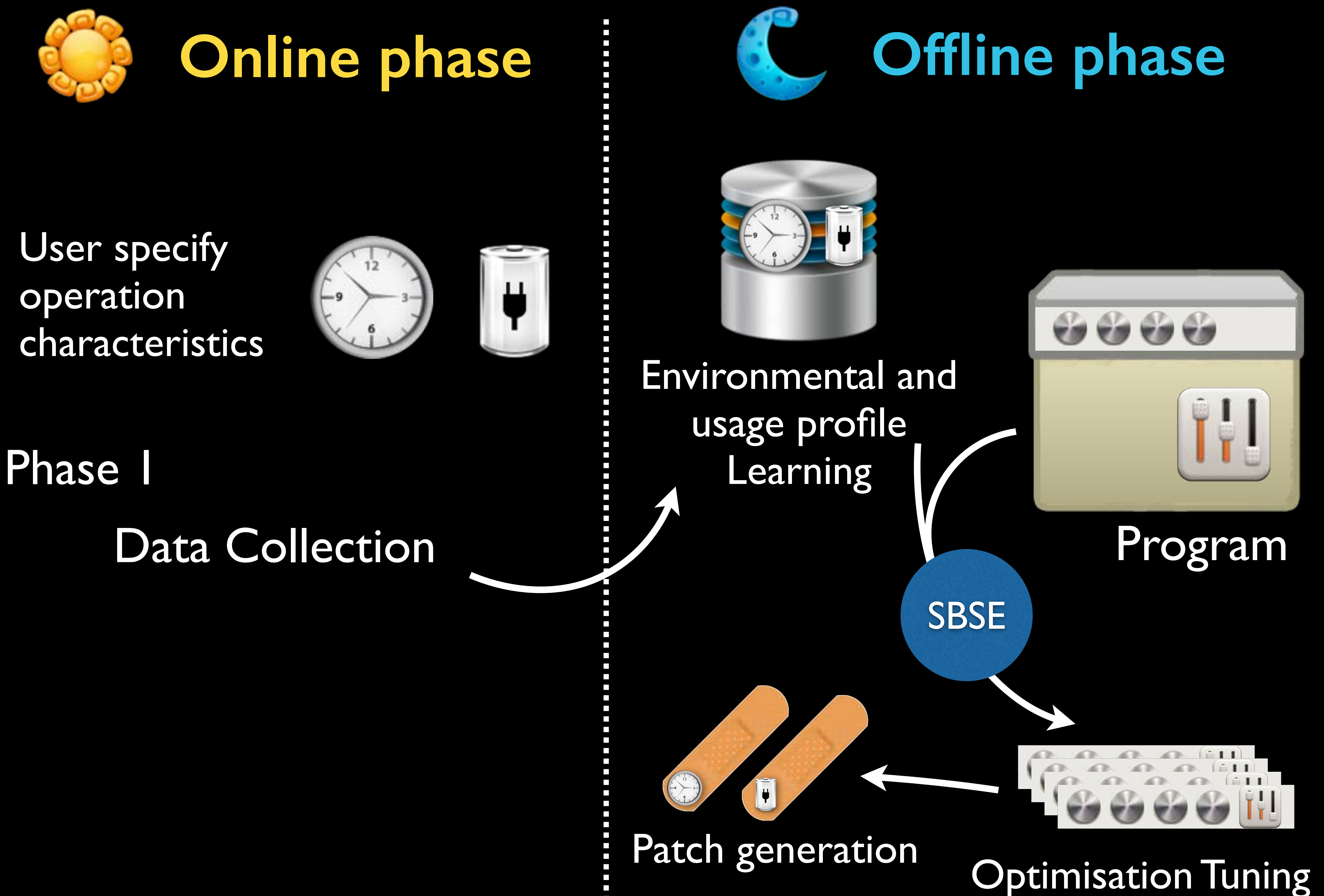
Tuneable Parameters

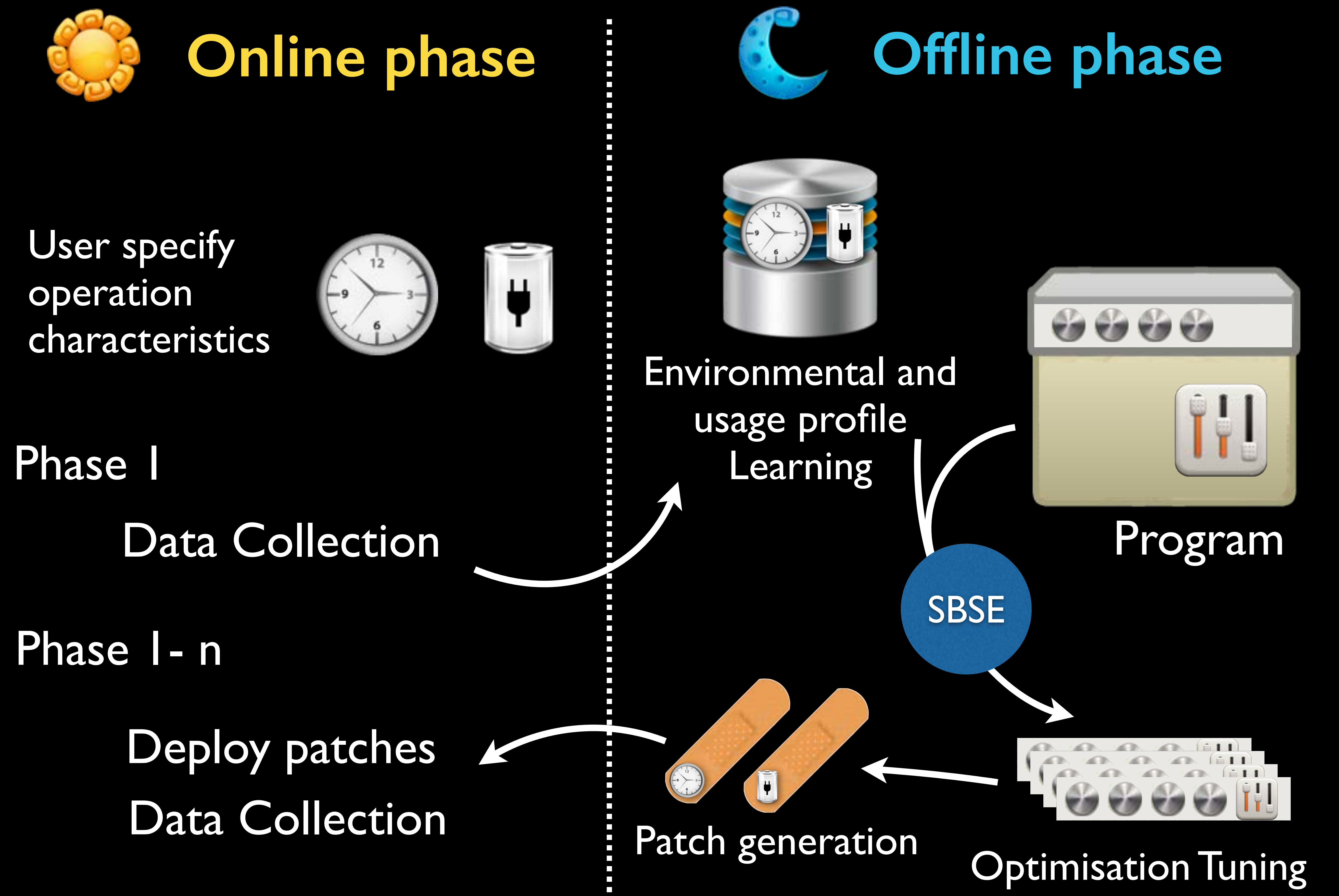
Program

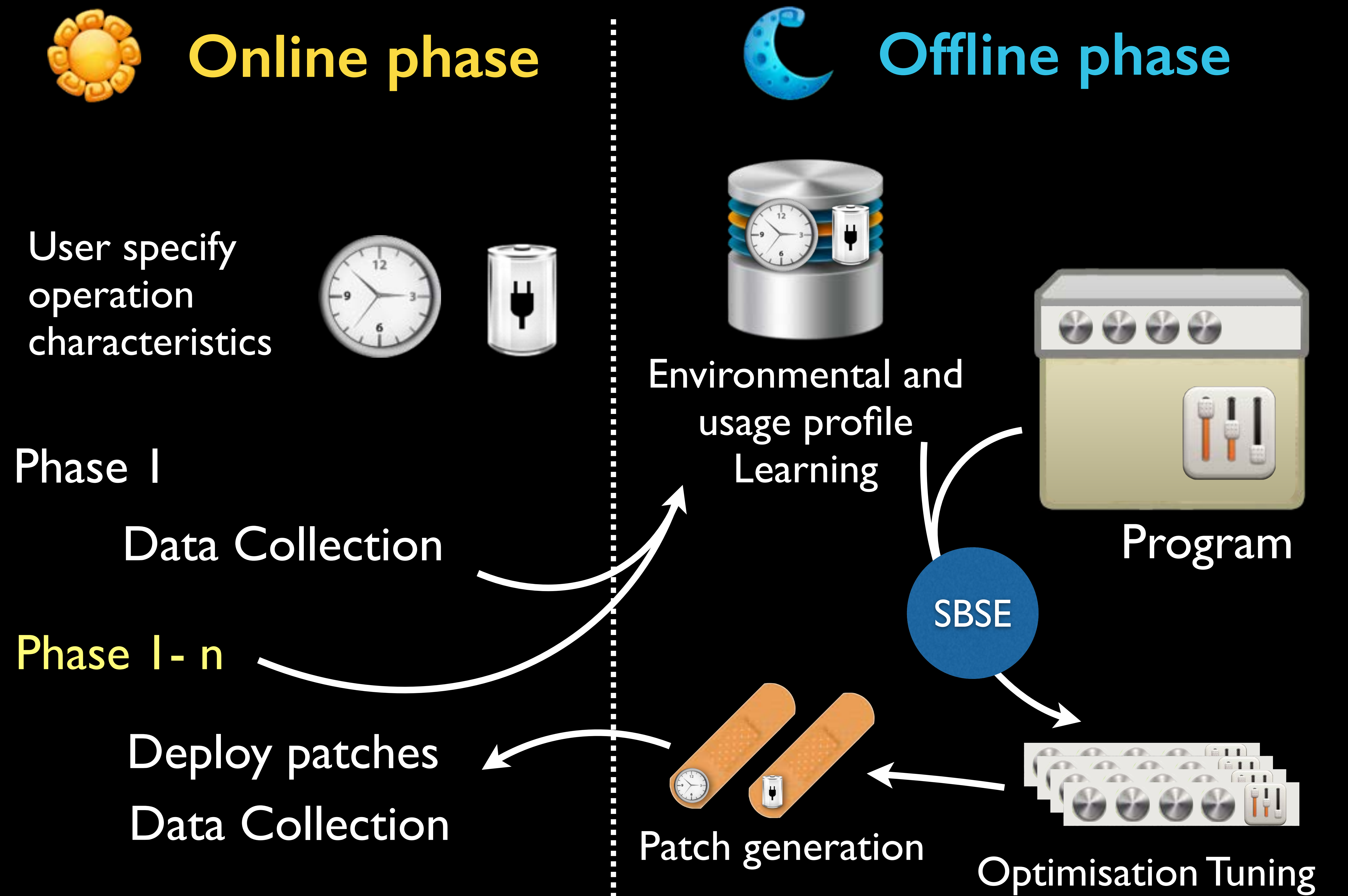
Tuneable Implicit Parameters

Exposing Implicit Parameters


Online phase
Offline phase
User specify operation characteristics
Phase 1
Data Collection
Environmental and usage profile Learning
Program
SBSE
Optimisation Tuning

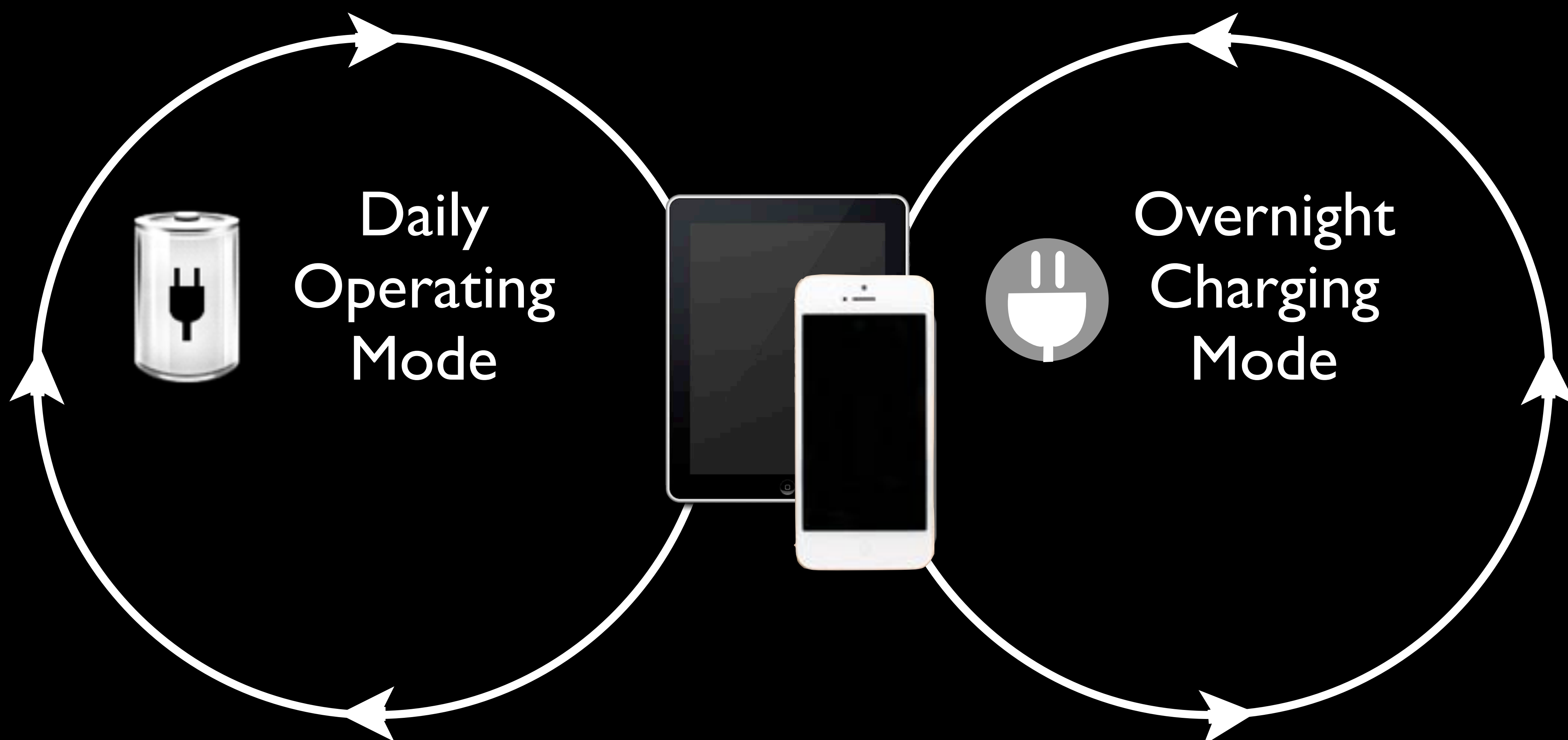

Online phase
Offline phase
User specify operation characteristics
Phase 1
Data Collection
Environmental and usage profile Learning
Program
SBSE
Patch generation
Optimisation Tuning


Online phase
Offline phase
User specify operation characteristics
Phase 1
Data Collection
Phase 1- n
Deploy patches
Data Collection
Environmental and usage profile Learning
Program
SBSE
Patch generation
Optimisation Tuning

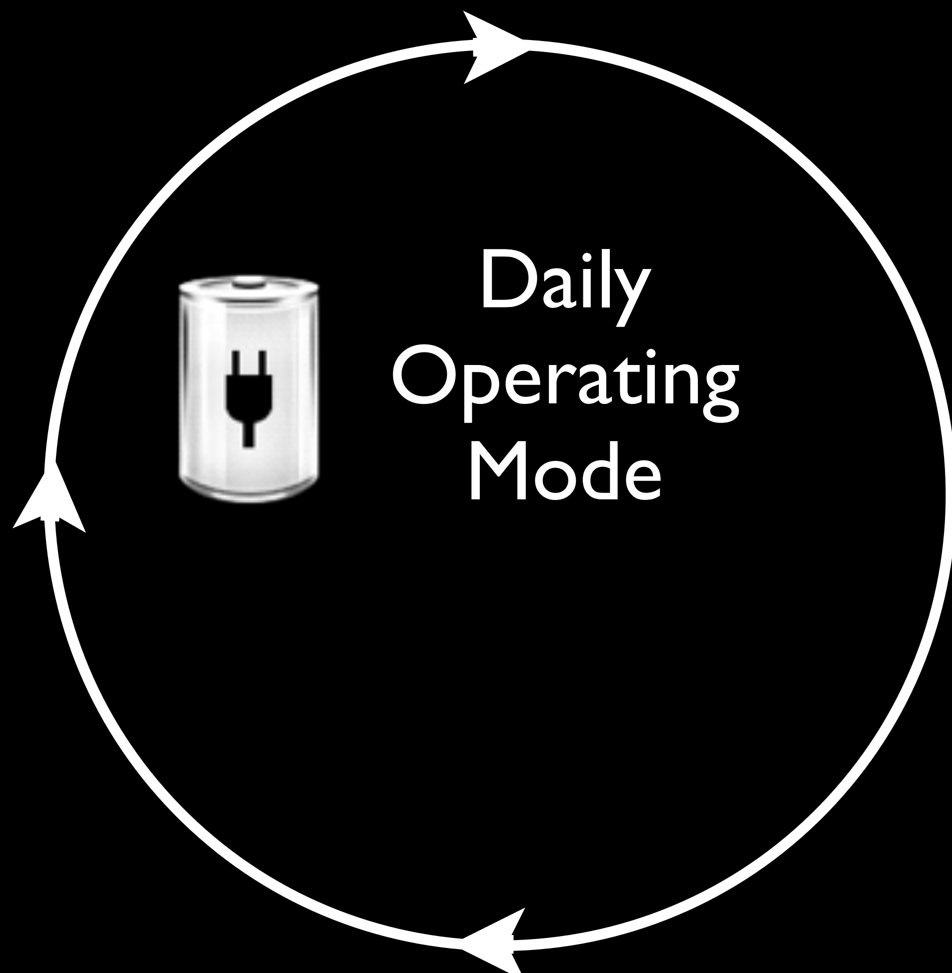

Online phase
Offline phase
User specify
operation
characteristics
Phase 1
Data Collection
Phase 1- n
Deploy patches
Data Collection
Environmental and
usage profile
Learning
Program
SBSE
Patch generation
Optimisation Tuning

Example: Wikipedia Mobile

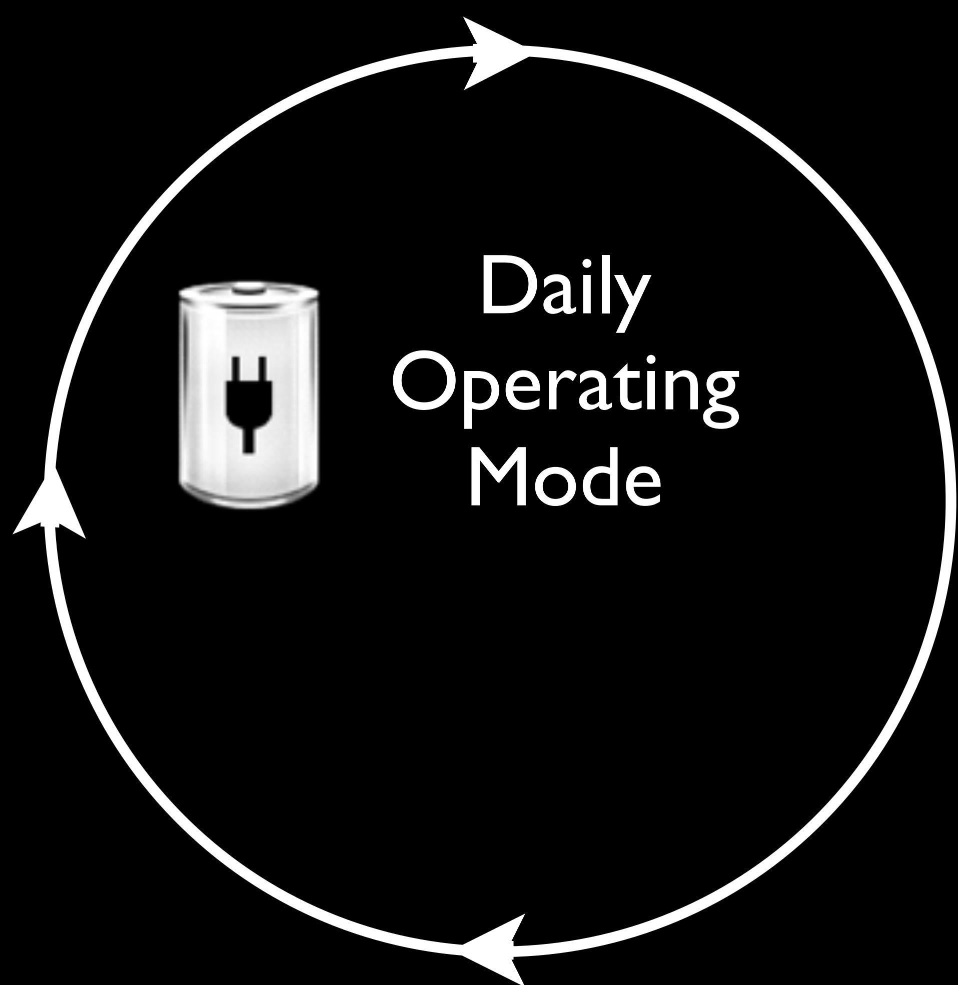
Example: Wikipedia Mobile



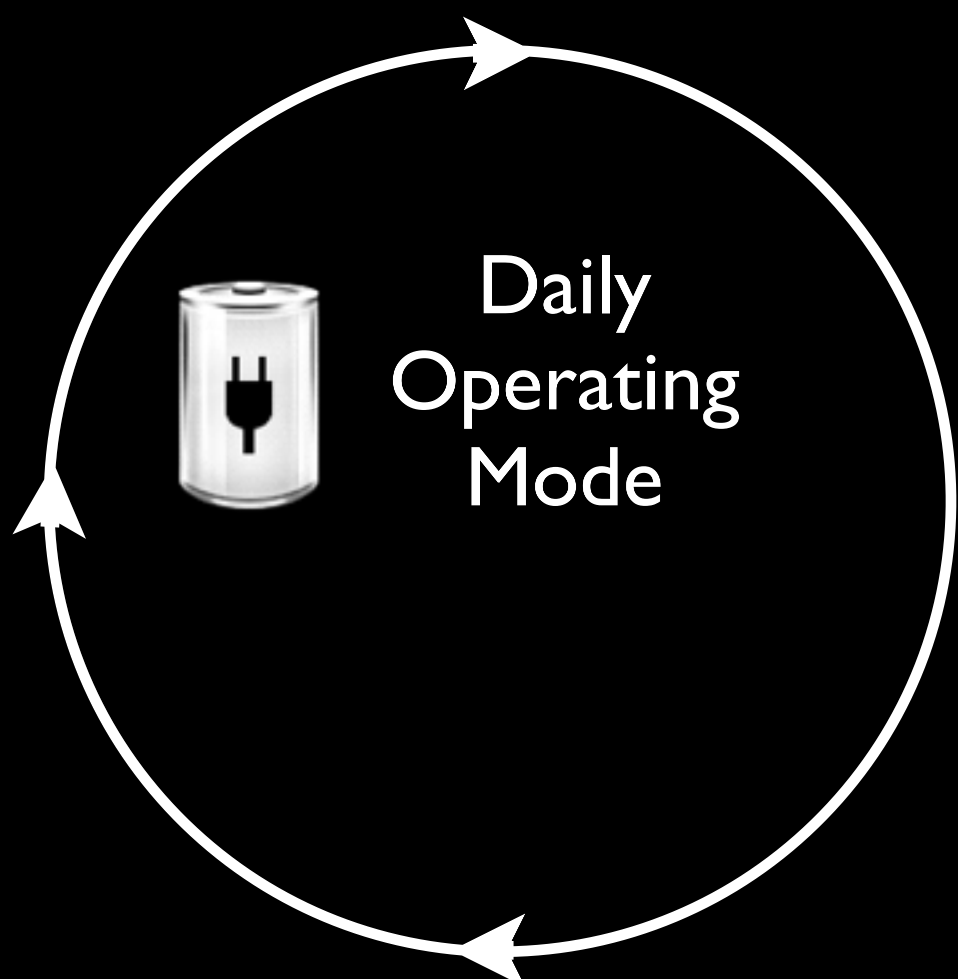
Example: Wikipedia Mobile



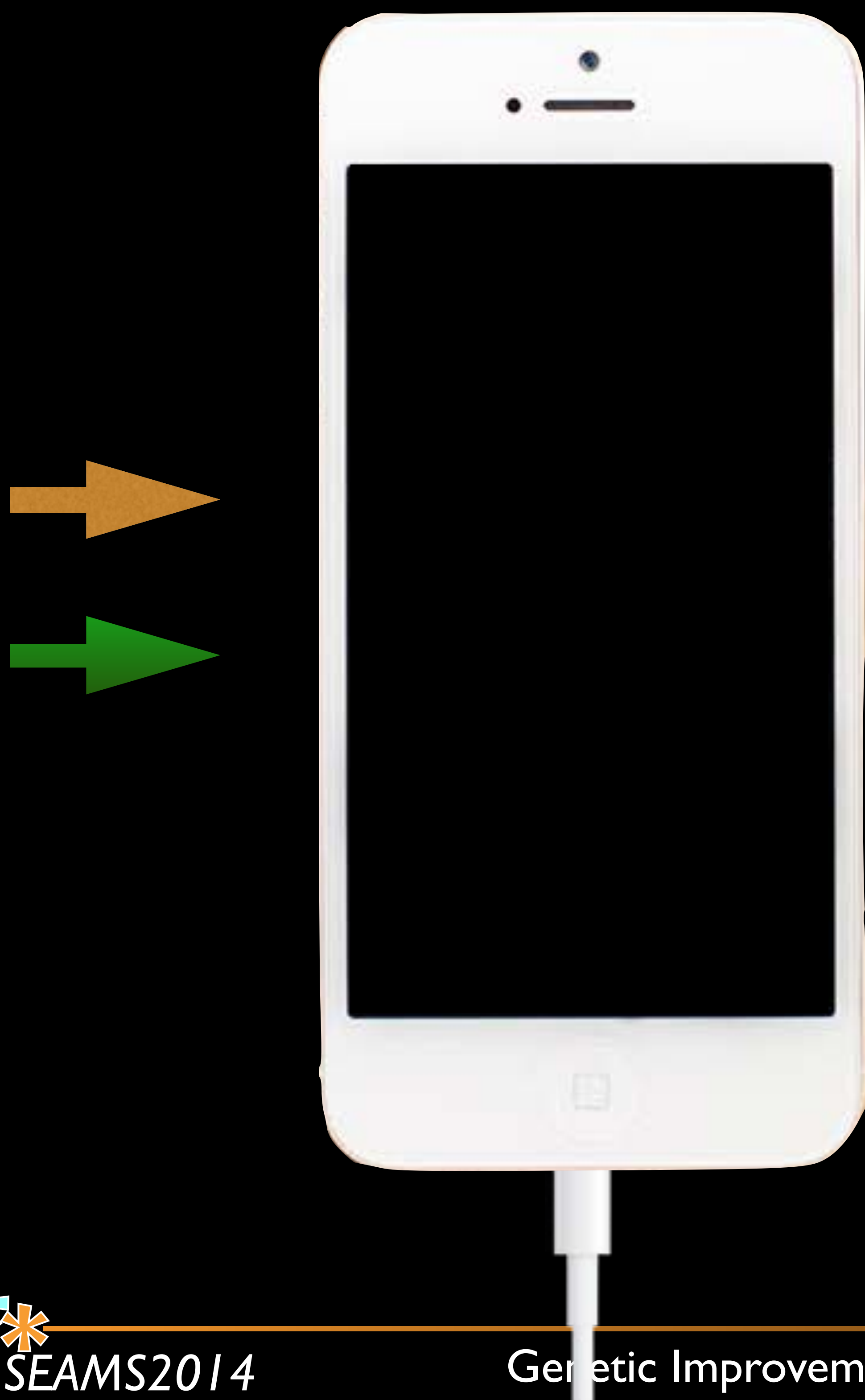
Example: Wikipedia Mobile

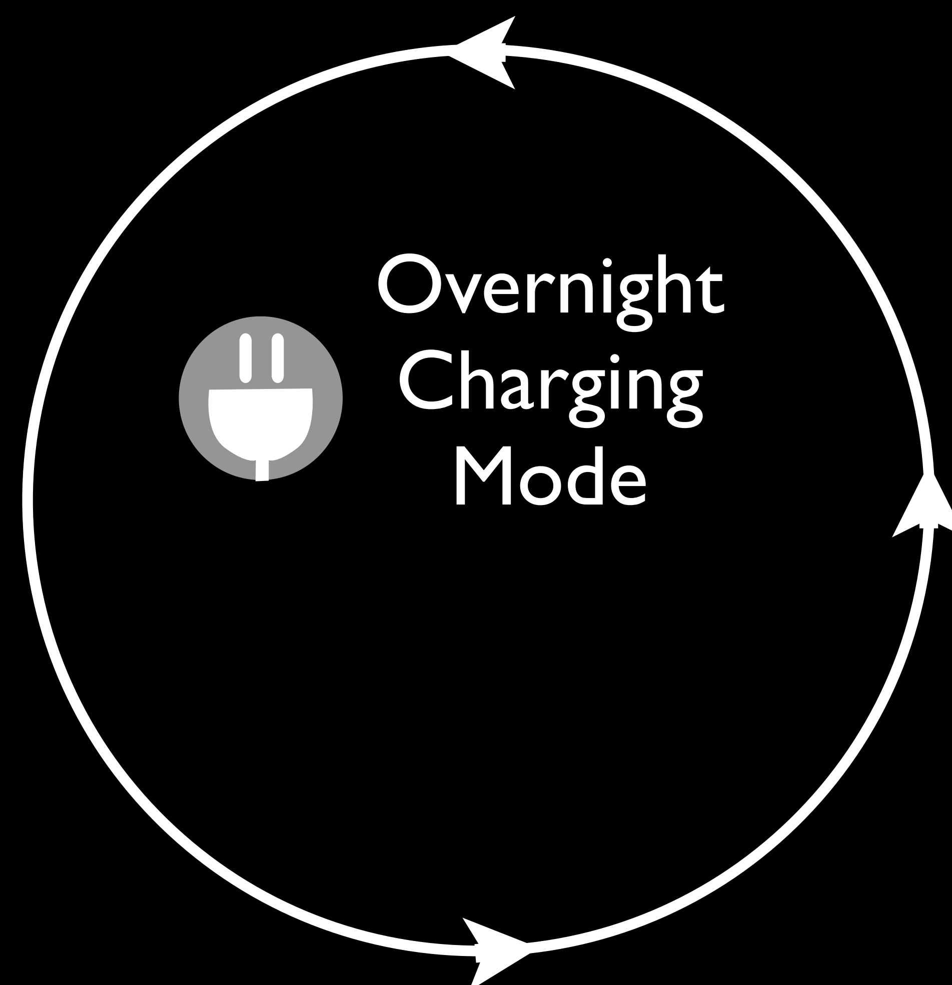


Example: Wikipedia Mobile

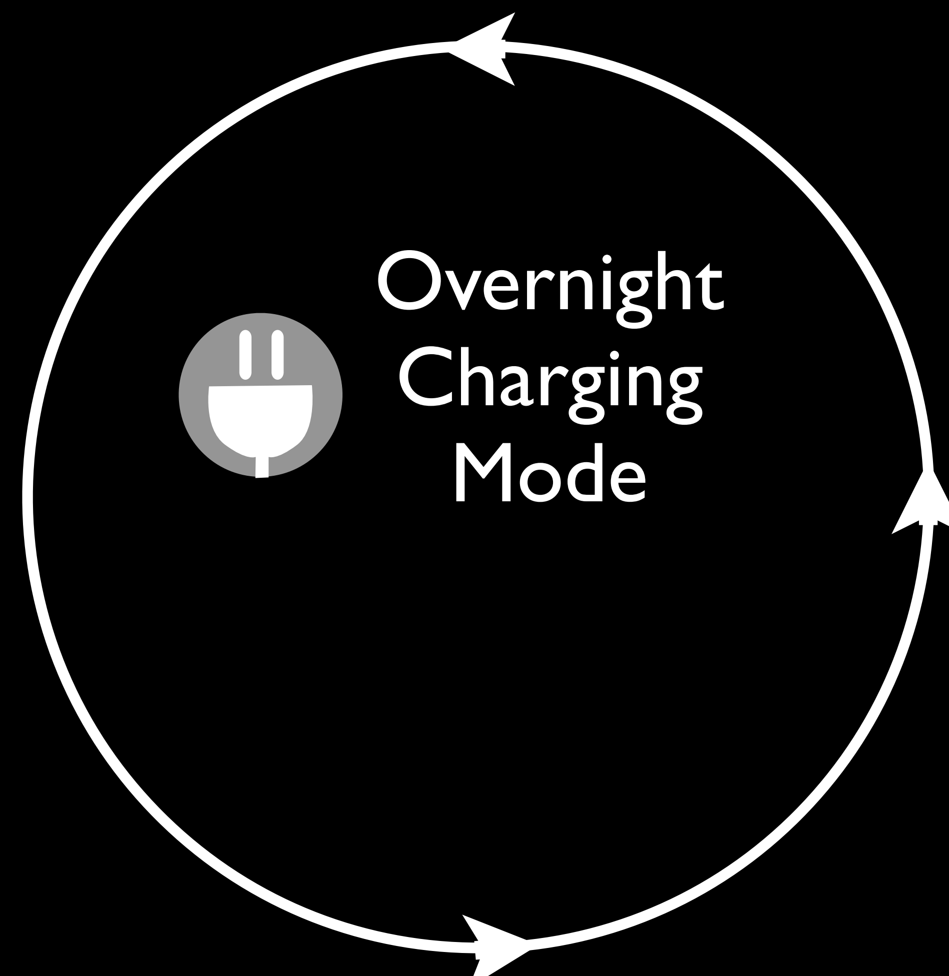


Example: Wikipedia Mobile





Example: Wikipedia Mobile



GP to search for caching strategies

Online Genetic Improvement

… could provide an *SBSE for Self Adaptive Systems*

Genetic Improvement
+
Grow and Graft Patches
+
Expose parameters and Autotune
+
Learning and deployment: Catch, Dream, Optimise

More details in keynote papers: ASE12, WCRE13 & SEAMS14