GI++ == Focused Auto Programming?

Robert Feldt

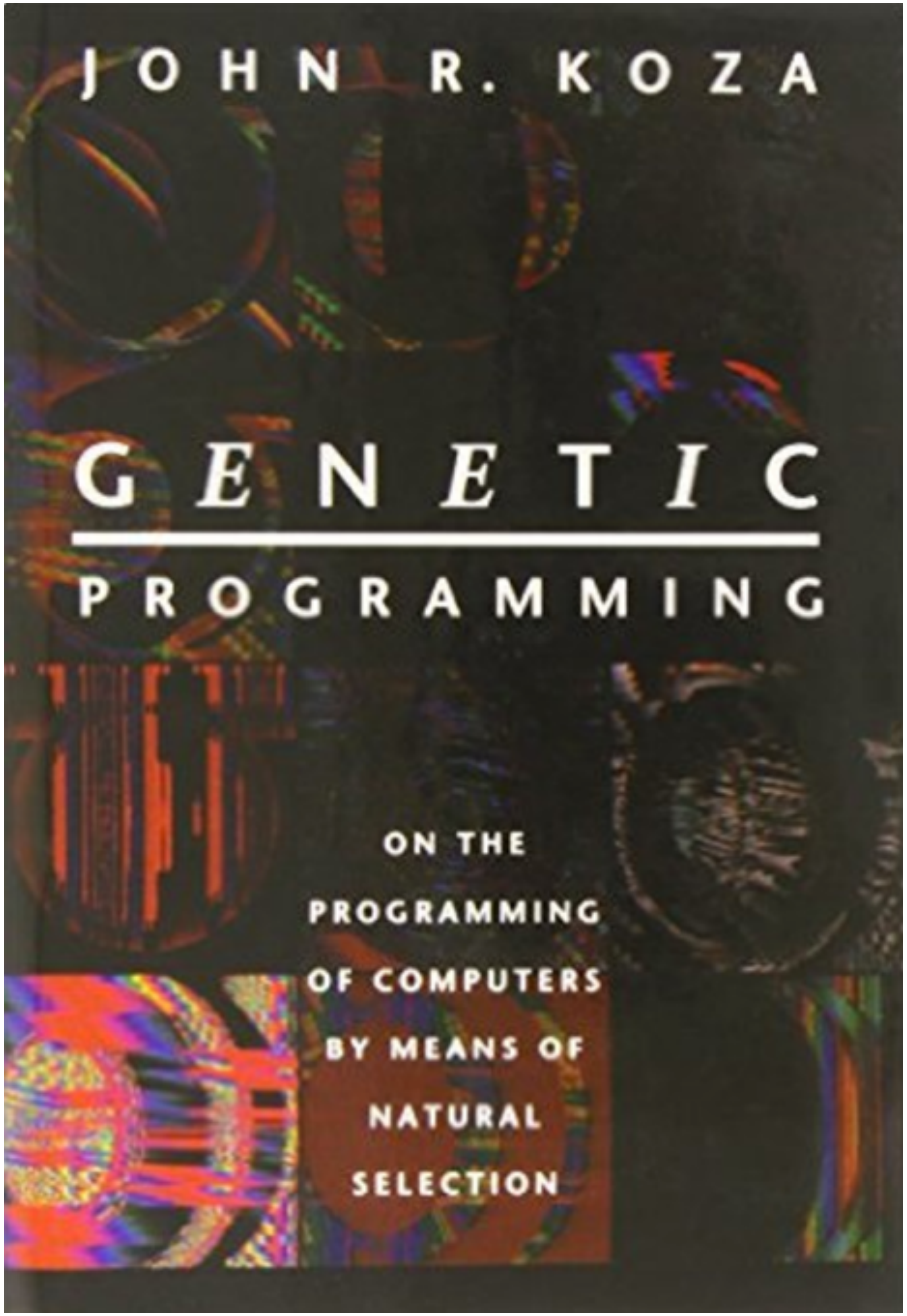
Chalmers University of Technology, Sweden

at the COW-50, UCL, London, 2017-01-31

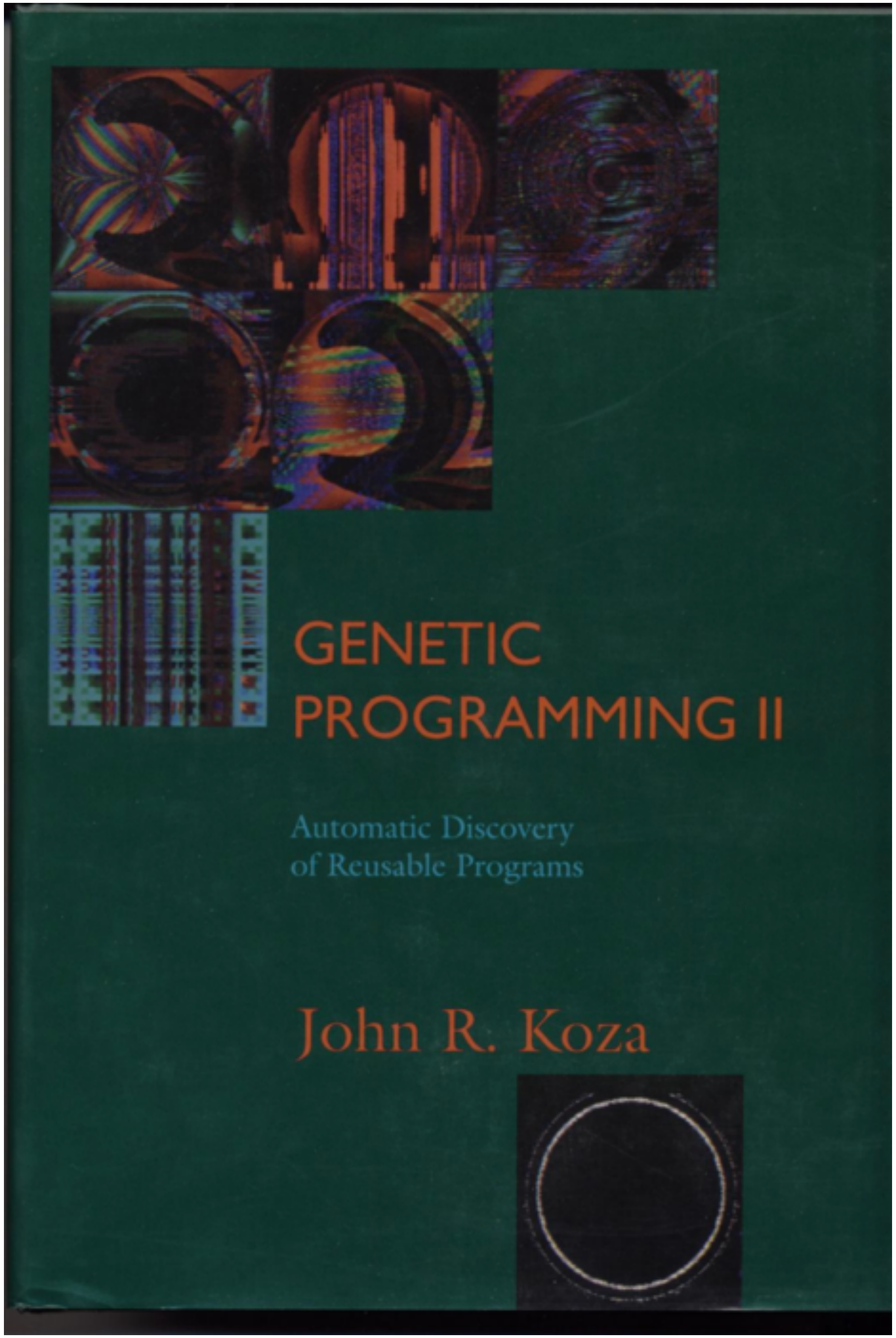
@drfeldt

One view of SBSE: Ever-expanding Success!

A contrarian view of SBSE: Not quite there yet…


JOHN R. KOZA
GENETIC
PROGRAMMING
ON THE PROGRAMMING OF COMPUTERS BY MEANS OF NATURAL SELECTION

A contrarian view of SBSE: Not quite there yet…

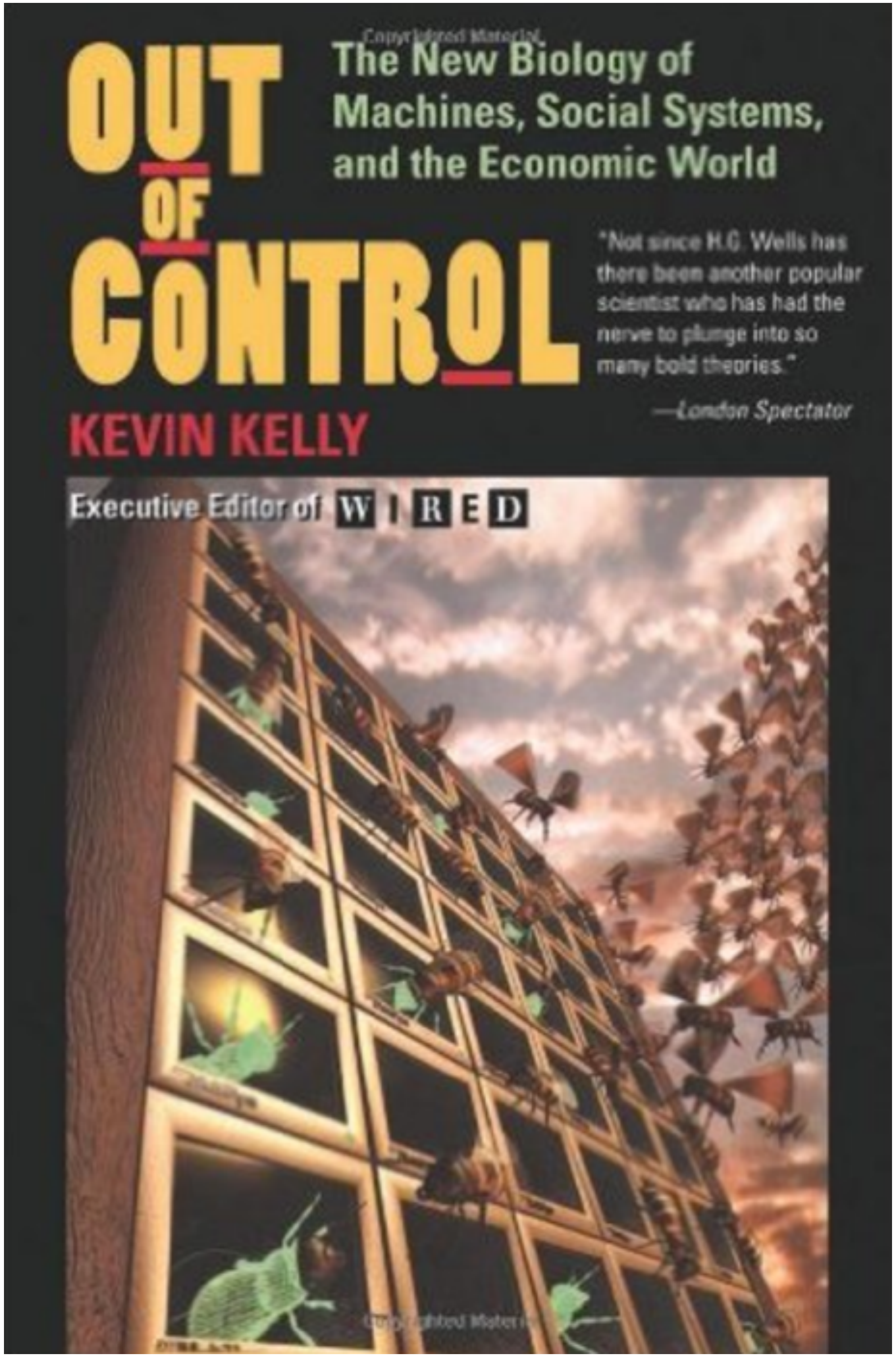


GENETIC PROGRAMMING II

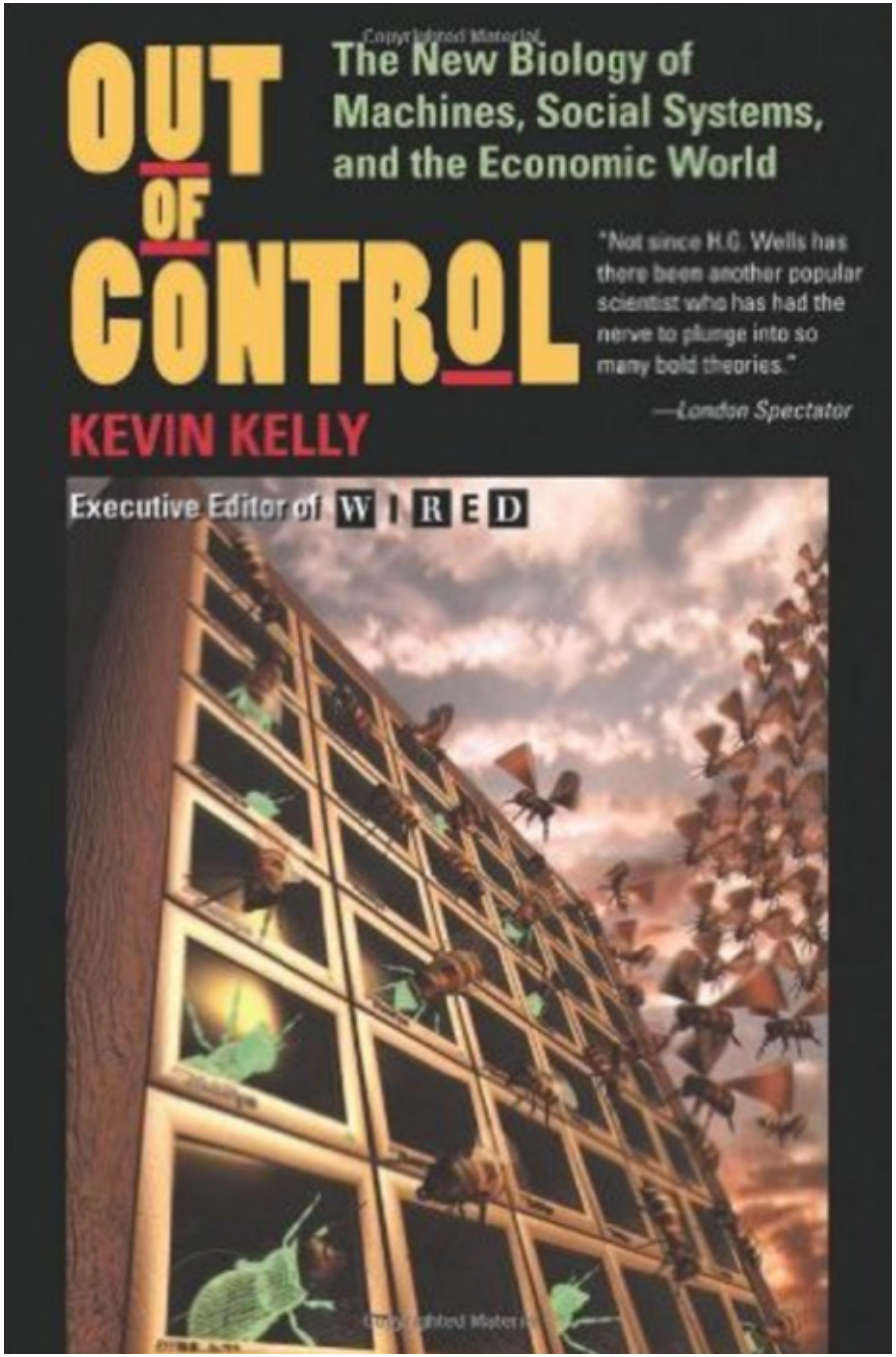
Automatic Discovery of Reusable Programs

John R. Koza

A contrarian view of SBSE: Not quite there yet…

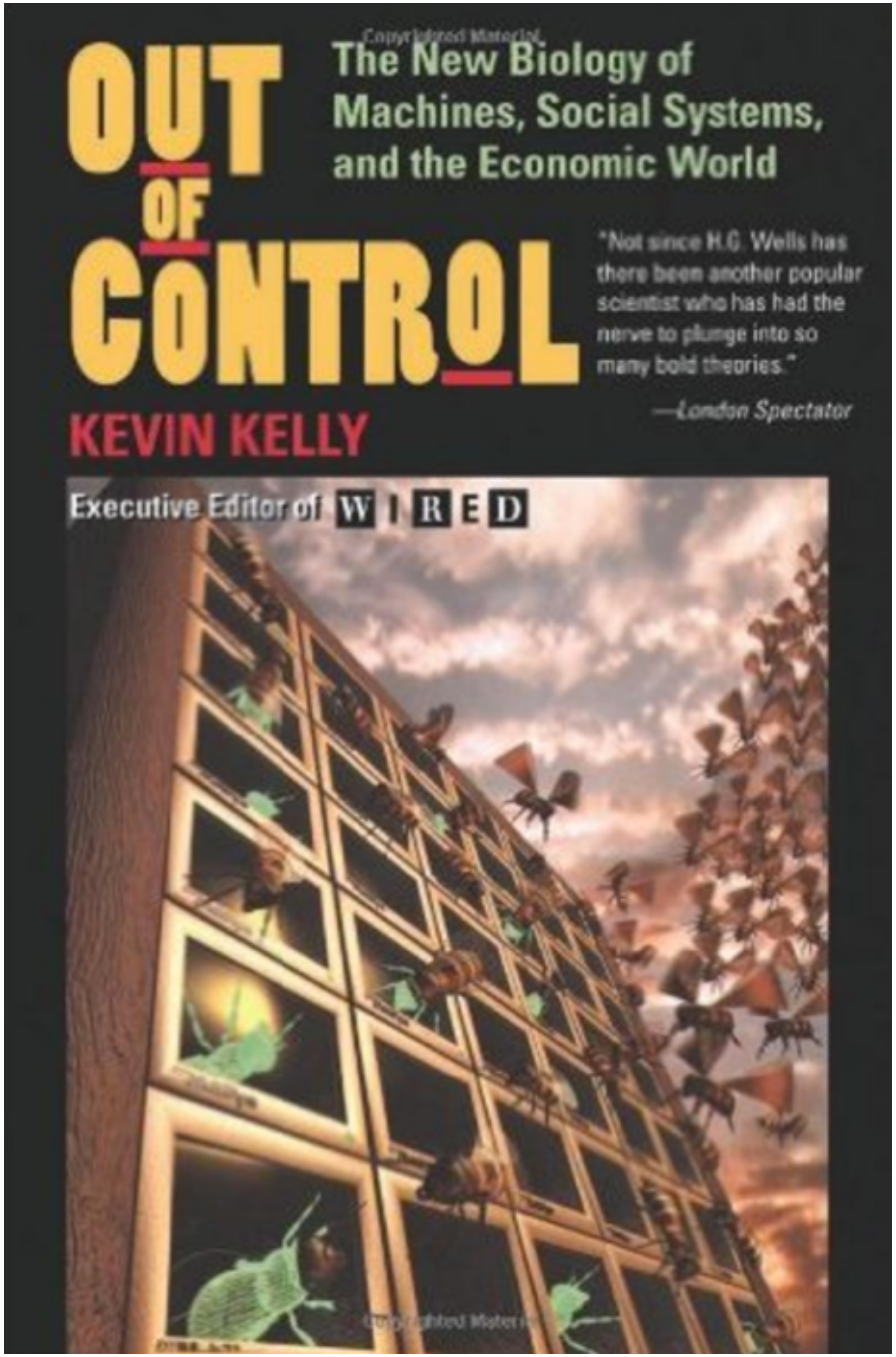

OUT OF CONTROL
The New Biology of Machines, Social Systems, and the Economic World
"Not since H.G. Wells has there been another popular scientist who has had the nerve to plunge into so many bold theories."
—London Spectator
KEVIN KELLY
Executive Editor of WIRED

A contrarian view of SBSE: Not quite there yet…


OUT OF CONTROL
The New Biology of Machines, Social Systems, and the Economic World
"Not since H.G. Wells has there been another popular scientist who has had the nerve to plunge into so many bold theories."
—London Spectator
KEVIN KELLY
Executive Editor of WIRED

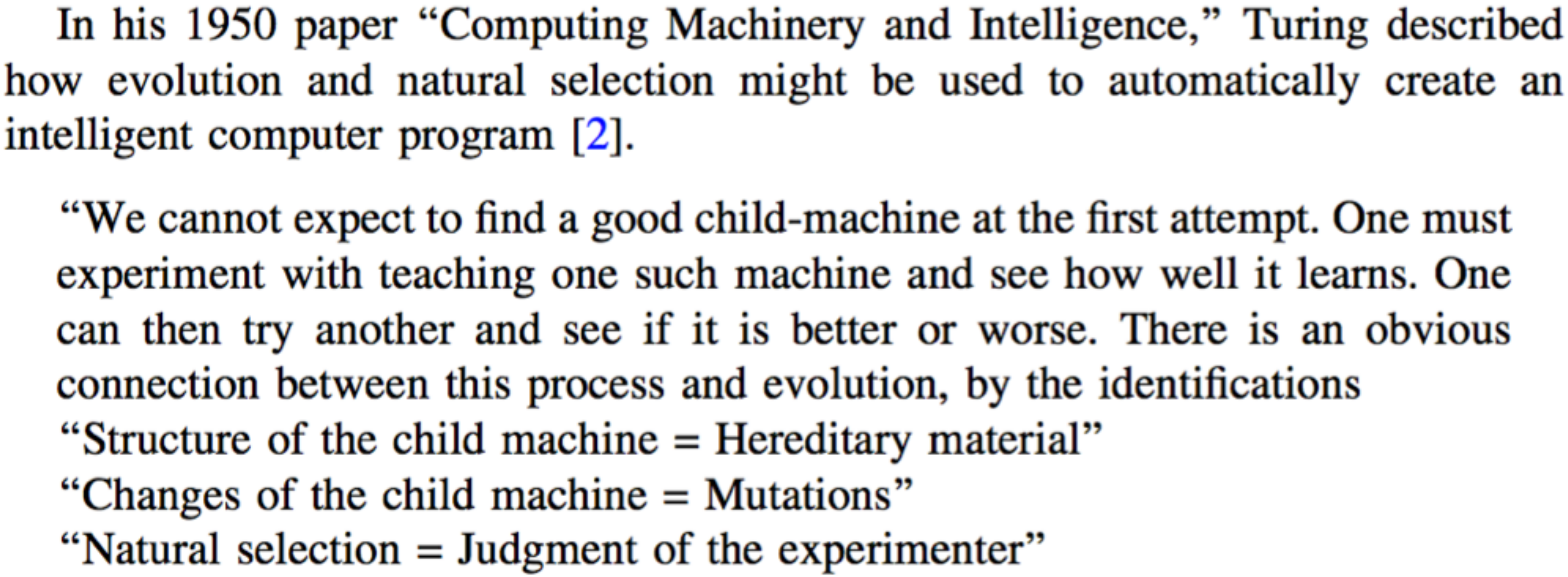
"Evolution is the natural way to program" - Tom Ray

A contrarian view of SBSE: Not quite there yet…


OUT OF CONTROL
The New Biology of Machines, Social Systems, and the Economic World
"Not since H.G. Wells has there been another popular scientist who has had the nerve to plunge into so many bold theories."
—London Spectator
KEVIN KELLY
Executive Editor of WIRED

"I would rather fly on a plane running software evolved by a program like this, than fly on a plane running software I wrote myself," says Hillis, programmer extraordinaire.

Of course it all started much earlier (with Turing)… ;)



In his 1950 paper “Computing Machinery and Intelligence,” Turing described how evolution and natural selection might be used to automatically create an intelligent computer program [2].

> “We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications
> “Structure of the child machine = Hereditary material”
> “Changes of the child machine = Mutations”
> “Natural selection = Judgment of the experimenter”

[Koza2010] in GPEM Anniversary issue

Some common GP/SBSE “cop outs”

Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program

Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code

Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code
- Focus on (minimal) interfaces between existing codes

Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code
- Focus on (minimal) interfaces between existing codes
- Focus on non-mainstream/obscure languages / processing formalisms where humans (currently) have less experience

Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code
- Focus on (minimal) interfaces between existing codes
- Focus on non-mainstream/obscure languages / processing formalisms where humans (currently) have less experience
- Evolve test data rather than programs

Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code
- Focus on (minimal) interfaces between existing codes
- Focus on non-mainstream/obscure languages / processing formalisms where humans (currently) have less experience
- Evolve test data rather than programs
- Evolve test cases and not programs

Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code
- Focus on (minimal) interfaces between existing codes
- Focus on non-mainstream/obscure languages / processing formalisms where humans (currently) have less experience
- Evolve test data rather than programs
- Evolve test cases and not programs
- Requiring lots and lots of example Input/Outputs

Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code
- Focus on (minimal) interfaces between existing codes
- Focus on non-mainstream/obscure languages / processing formalisms where humans (currently) have less experience
- Evolve test data rather than programs
- Evolve test cases and not programs
- Requiring lots and lots of example Input/Outputs
- …

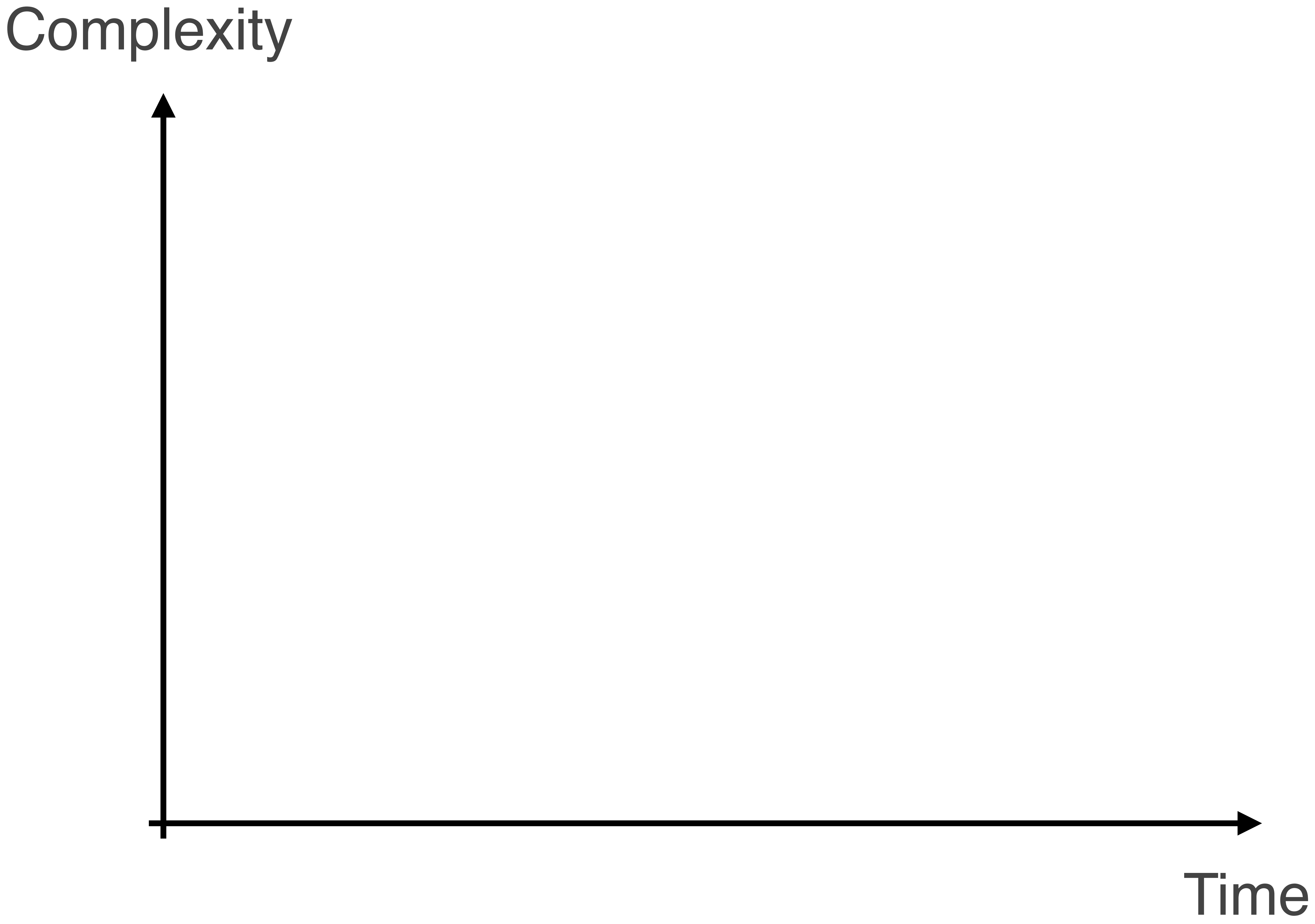
Some common GP/SBSE “cop outs”

- Tune only constants/numbers in fixed program
- Delete/remix existing code
- Focus on (minimal) interfaces between existing codes
- Focus on non-mainstream/obscure languages / processing formalisms where humans (currently) have less experience
- Evolve test data rather than programs
- Evolve test cases and not programs
- Requiring lots and lots of example Input/Outputs
- …

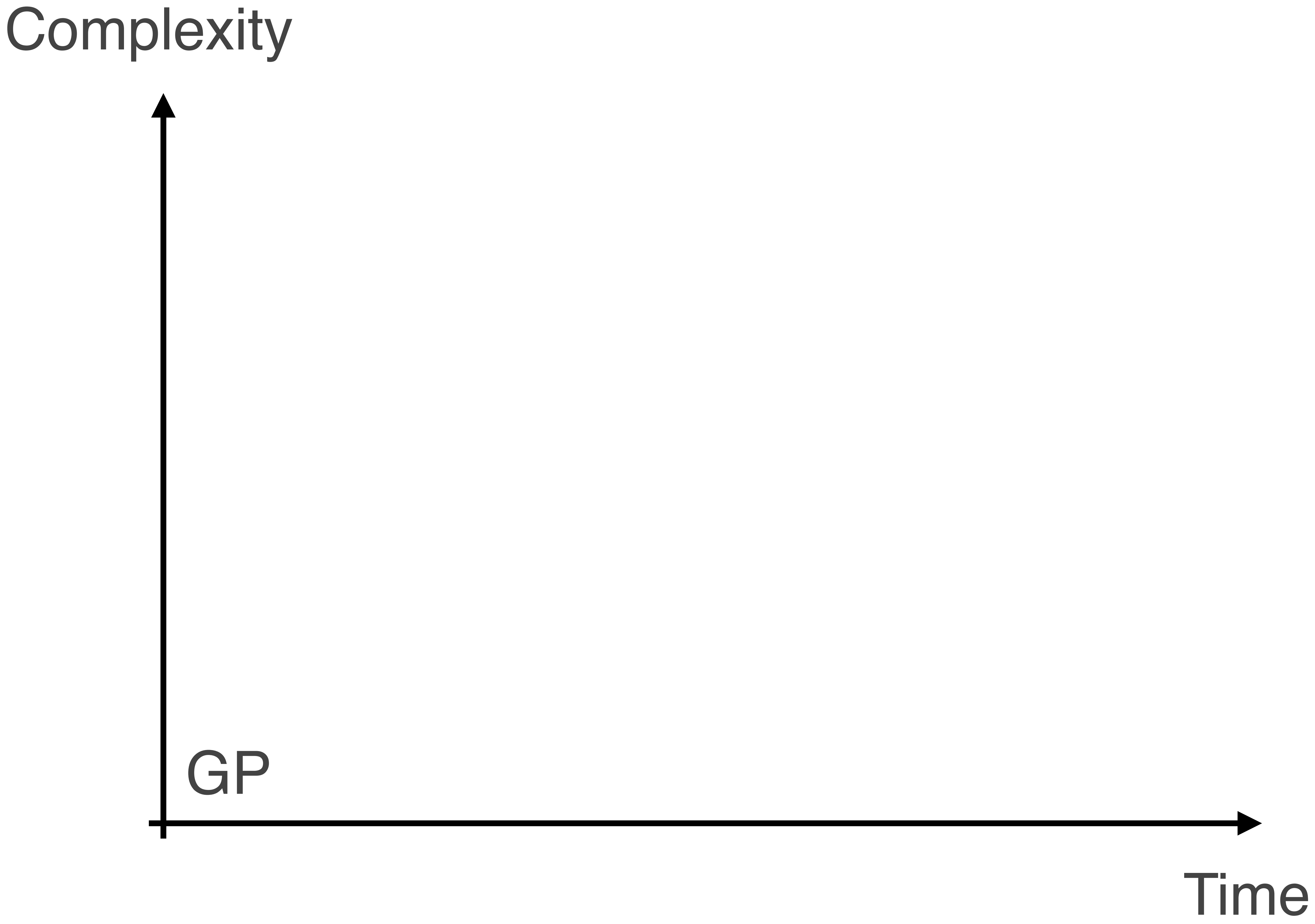
Clear goal, small search space, less/short structure

A continuum of Automated Programming

A continuum of Automated Programming



A continuum of Automated Programming

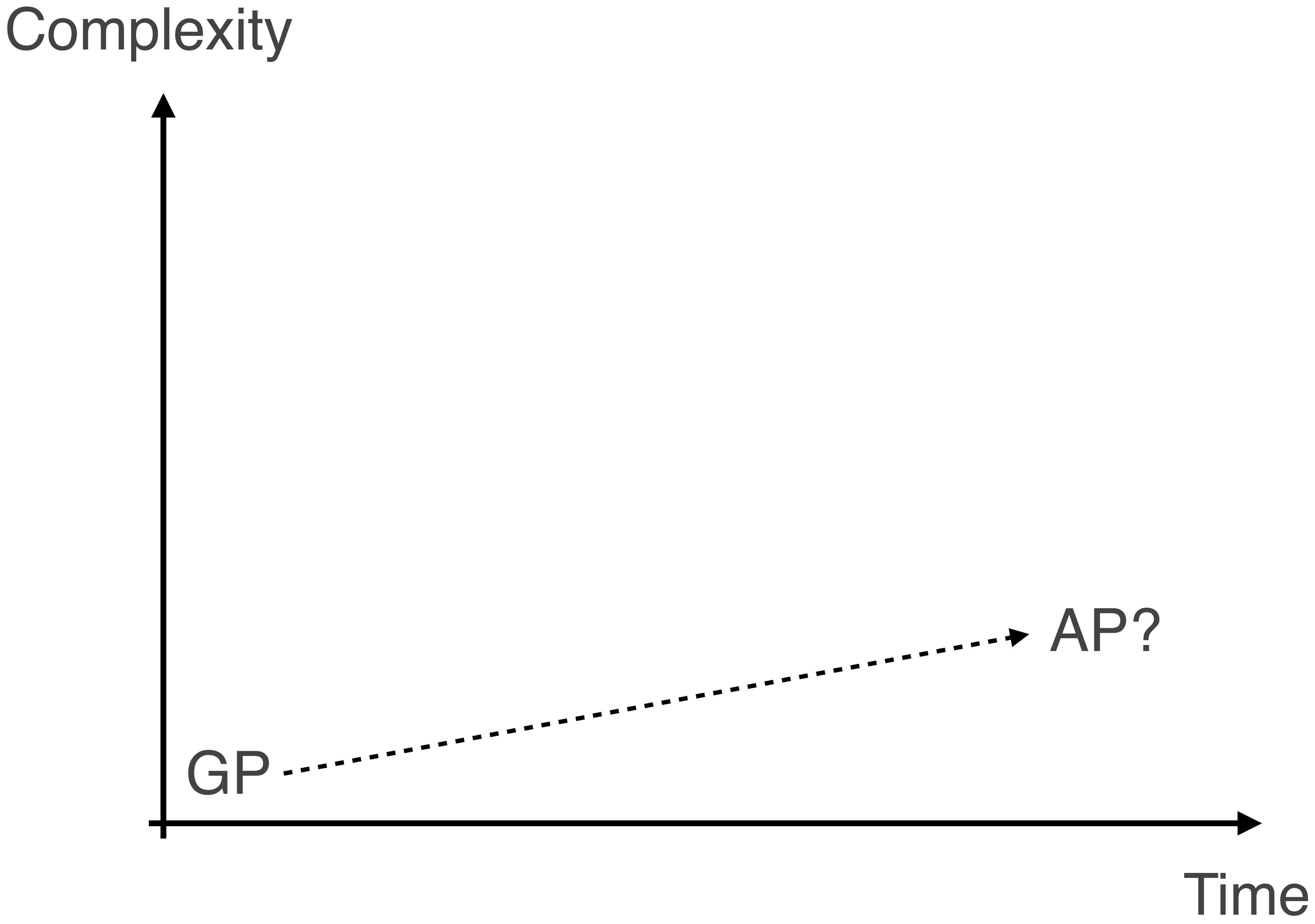


Complexity

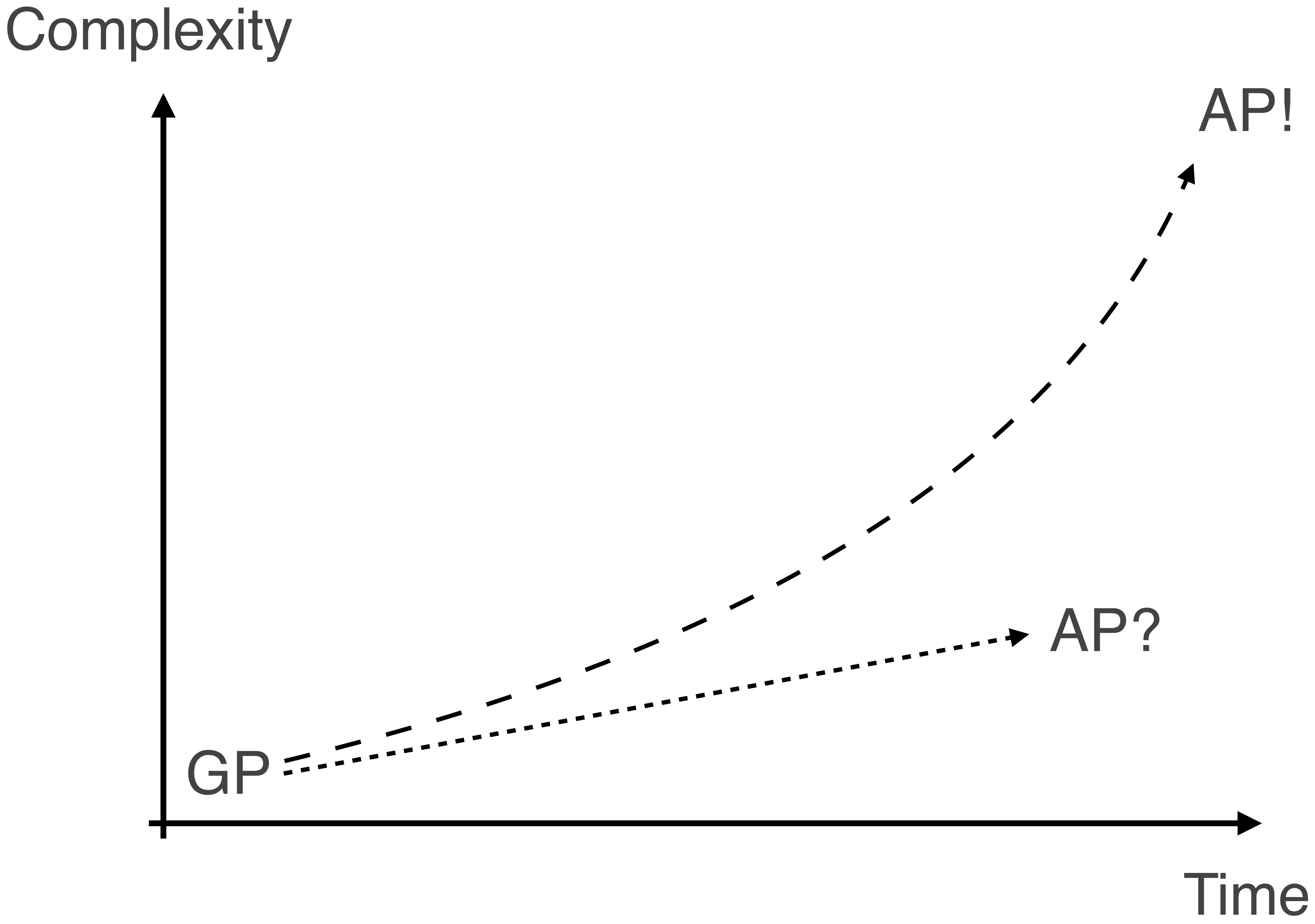
GP

Time

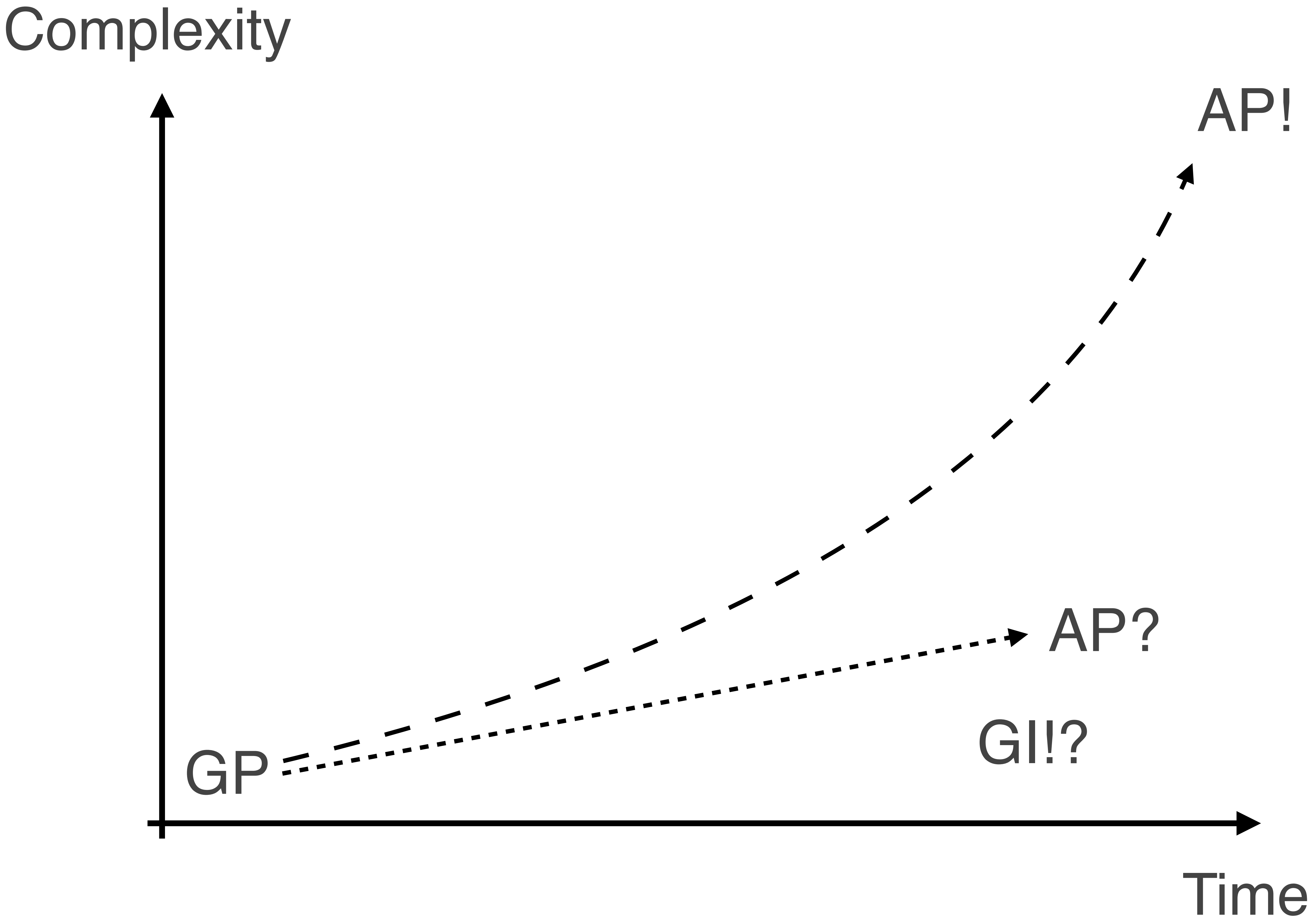
A continuum of Automated Programming



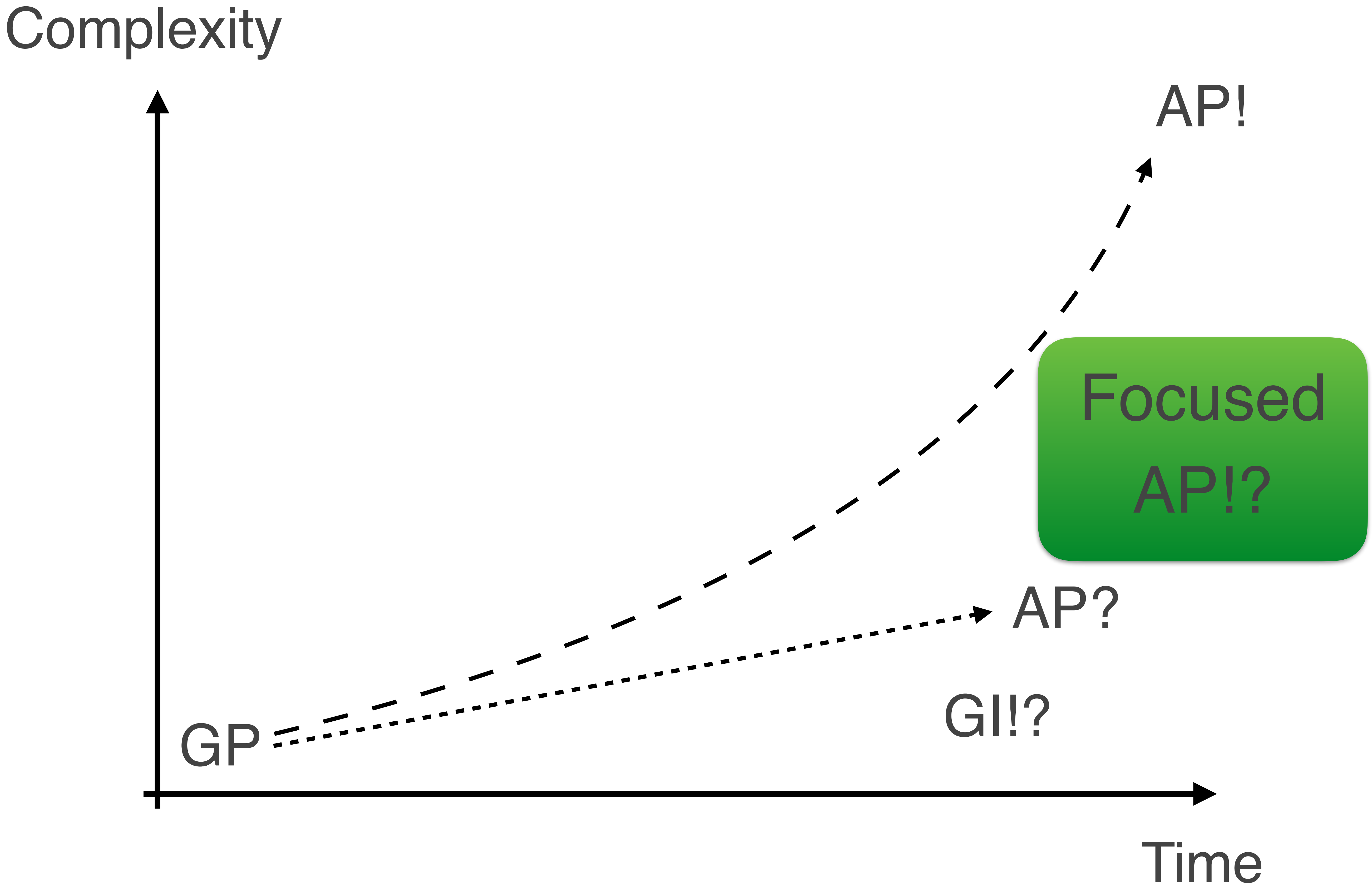
A continuum of Automated Programming



A continuum of Automated Programming



A continuum of Automated Programming



Focused Automated Programming

Focused Automated Programming

- I propose we should study FAP! aka…

Focused Automated Programming

- I propose we should study FAP! aka…
  - Domain-specific Automated Programming (DAP)

Focused Automated Programming

- I propose we should study FAP! aka…
  - Domain-specific Automated Programming (DAP)
  - Task-specific Automated Programming (TAP)

Focused Automated Programming

- I propose we should study FAP! aka…
  - Domain-specific Automated Programming (DAP)
  - Task-specific Automated Programming (TAP)
- Defined as: *“Focused application of search and optimisation to create/adapt/tune (parts of) program code during its development, setup and/or execution”*

Focused Automated Programming

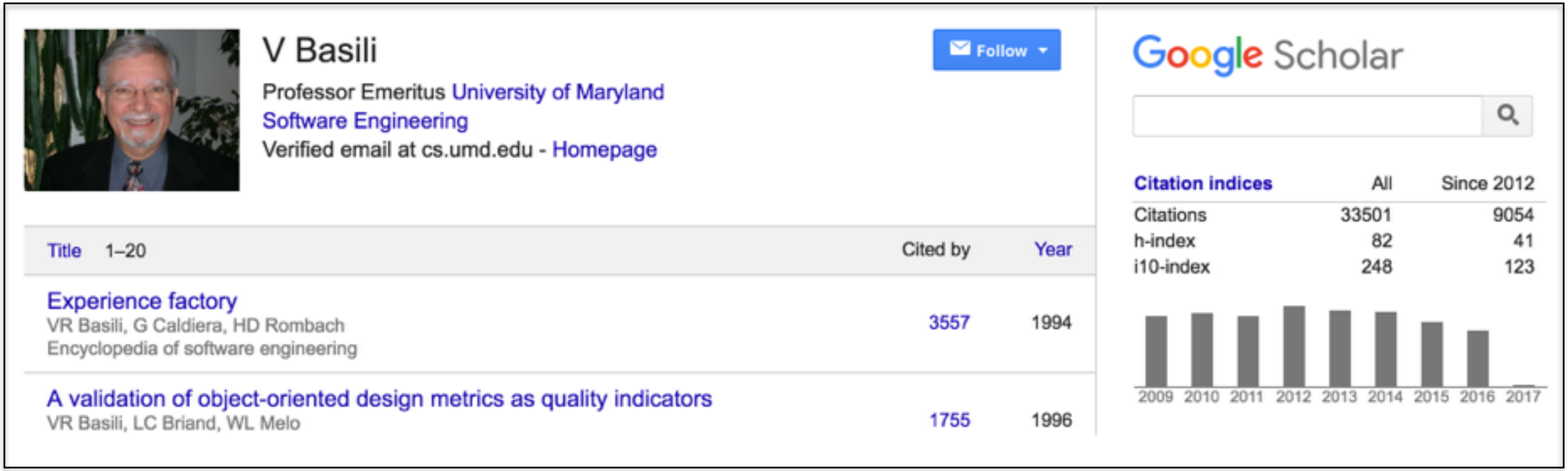
- I propose we should study FAP! aka…
  - Domain-specific Automated Programming (DAP)
  - Task-specific Automated Programming (TAP)
- Defined as: *“Focused application of search and optimisation to create/adapt/tune (parts of) program code during its development, setup and/or execution”*
- Focused here essentially means “human-guided”, i.e. it is a hybrid/interactive development philosophy

Focused Automated Programming

- I propose we should study FAP! aka…
  - Domain-specific Automated Programming (DAP)
  - Task-specific Automated Programming (TAP)
- Defined as: *“Focused application of search and optimisation to create/adapt/tune (parts of) program code during its development, setup and/or execution”*
- Focused here essentially means “human-guided”, i.e. it is a hybrid/interactive development philosophy
- => we need ideas, intuition and methods/processes for how to use search/optimisation more actively in the software development process

Example: Web extraction library

Example: Web extraction library



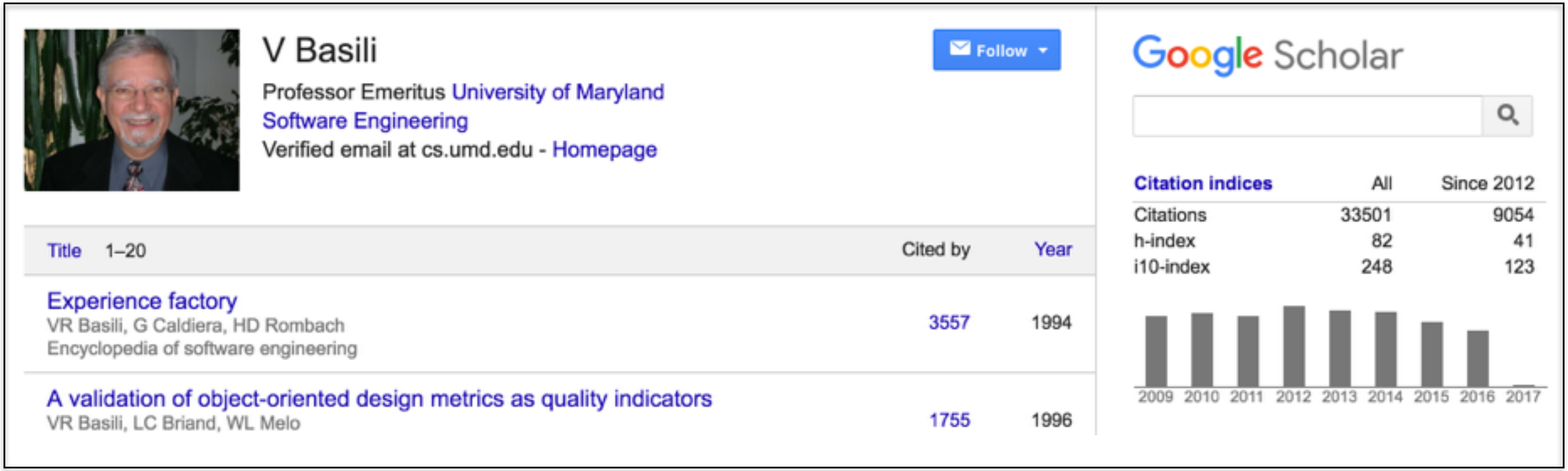
V Basili

Professor Emeritus University of Maryland
Software Engineering
Verified email at cs.umd.edu - Homepage

Title 1–20	Cited by	Year
Experience factory VR Basili, G Caldiera, HD Rombach Encyclopedia of software engineering	3557	1994
A validation of object-oriented design metrics as quality indicators VR Basili, LC Briand, WL Melo	1755	1996

Citation indices	All	Since 2012
Citations	33501	9054
h-index	82	41
i10-index	248	123

Example: Web extraction library



V Basili

Professor Emeritus University of Maryland
Software Engineering
Verified email at cs.umd.edu - Homepage

Title 1–20	Cited by	Year
Experience factory VR Basili, G Caldiera, HD Rombach Encyclopedia of software engineering	3557	1994
A validation of object-oriented design metrics as quality indicators VR Basili, LC Briand, WL Melo	1755	1996

Google Scholar

Citation indices	All	Since 2012
Citations	33501	9054
h-index	82	41
i10-index	248	123

```
{
   “name”: “V Basili”,
   “citations”: 33501,
   “h-index”: 82
}
```

Web extraction, traditional solution vs AdaptiLib

Web extraction, traditional solution vs AdaptiLib

WebGet
Lib

Web extraction, traditional solution vs AdaptiLib

WebGet Lib +

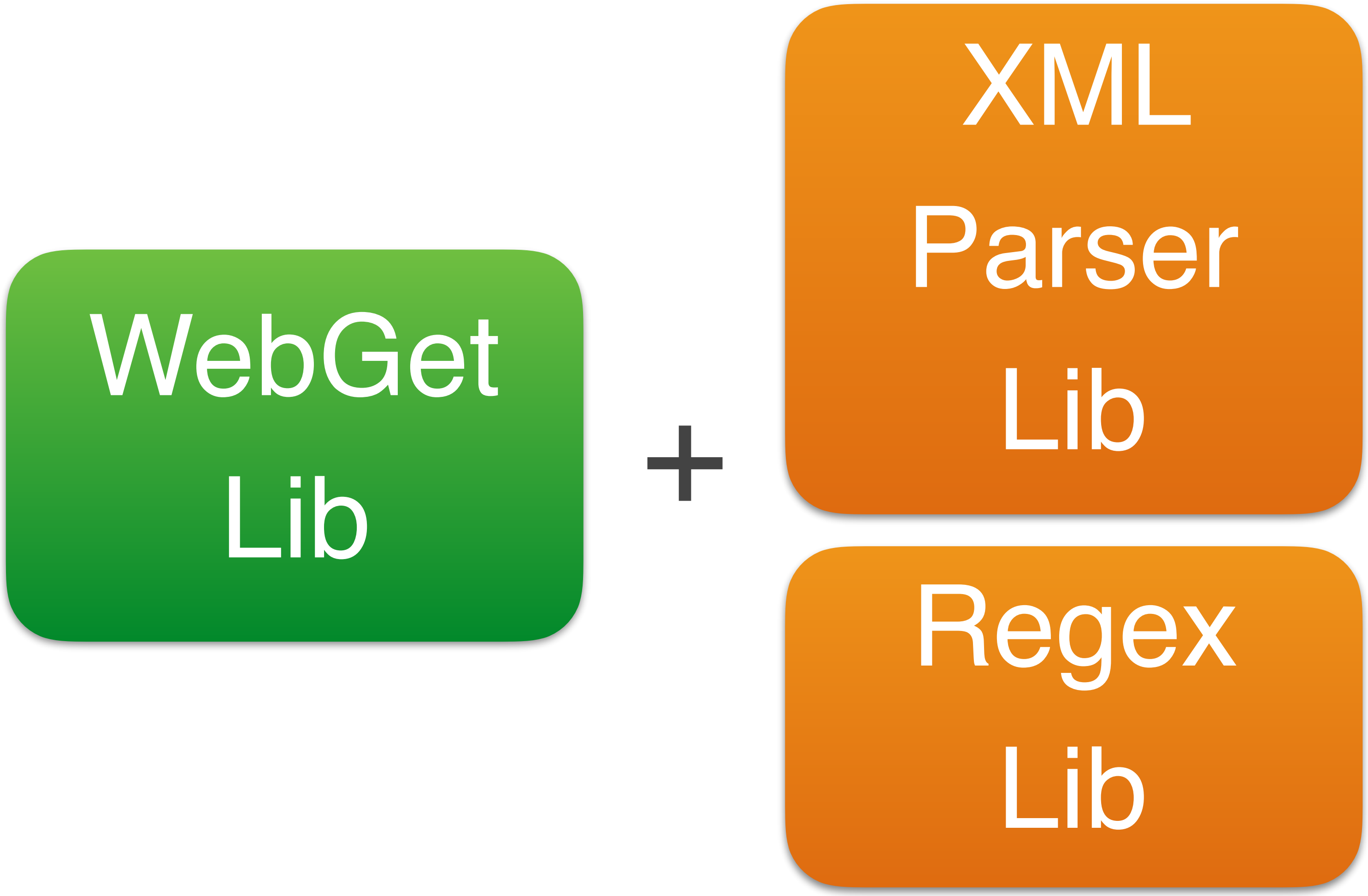
Web extraction, traditional solution vs AdaptiLib

WebGet Lib

+

XML Parser Lib

Web extraction, traditional solution vs AdaptiLib



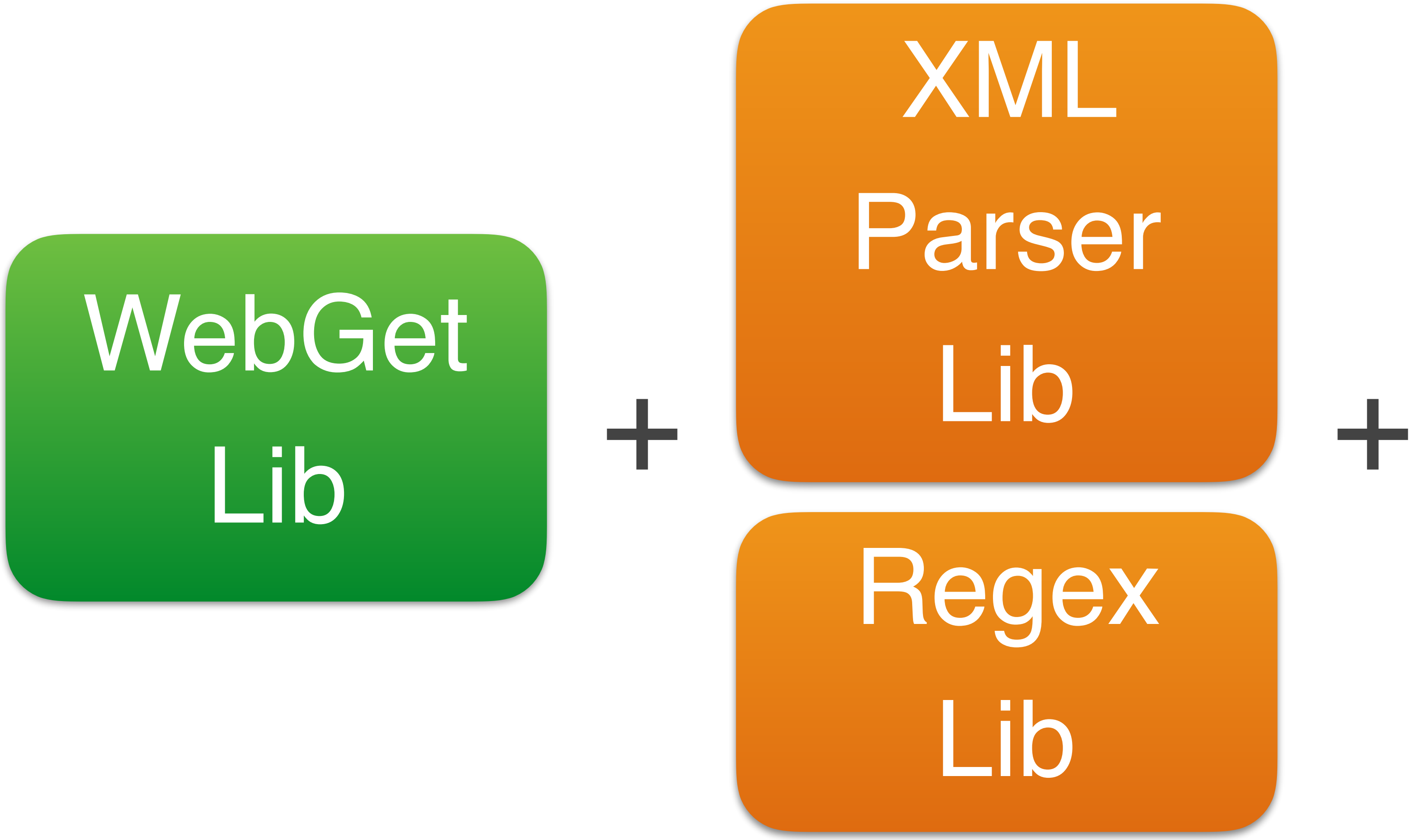
WebGet Lib

+

XML Parser Lib

Regex Lib

Web extraction, traditional solution vs AdaptiLib



WebGet Lib

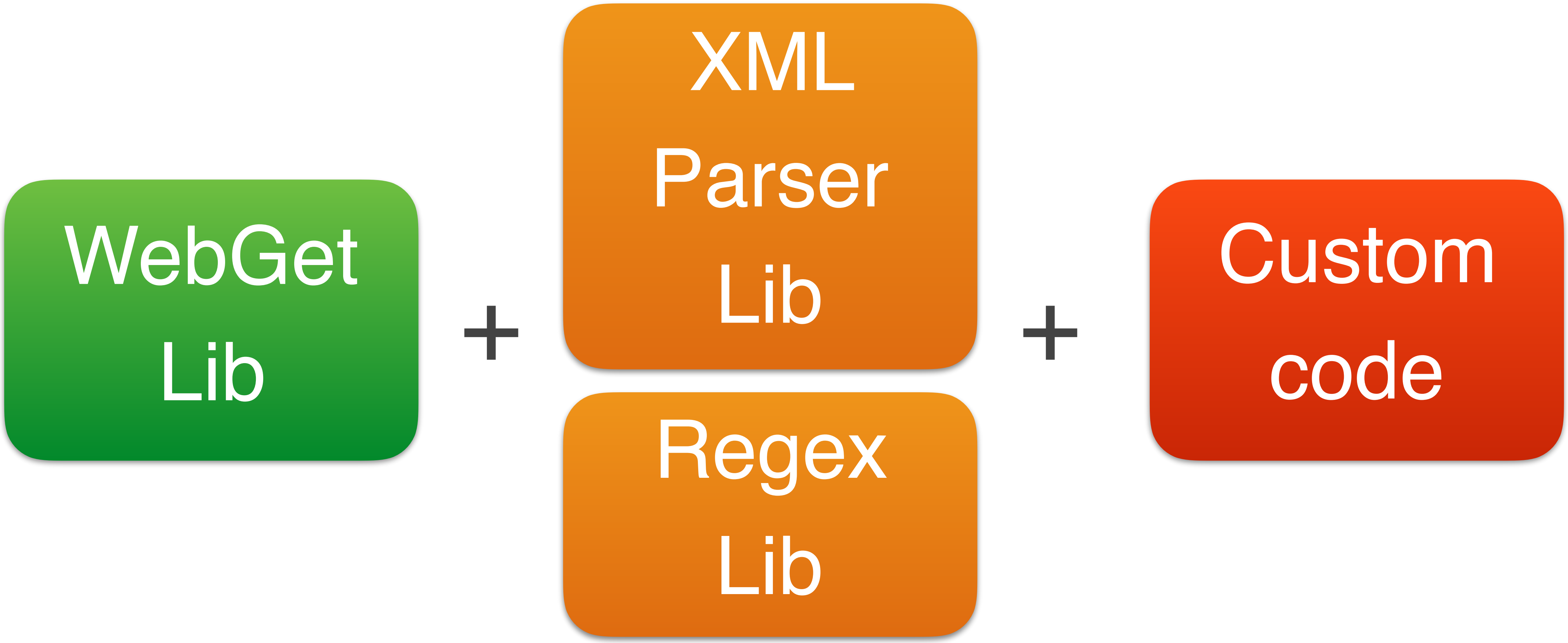
+

XML Parser Lib

Regex Lib

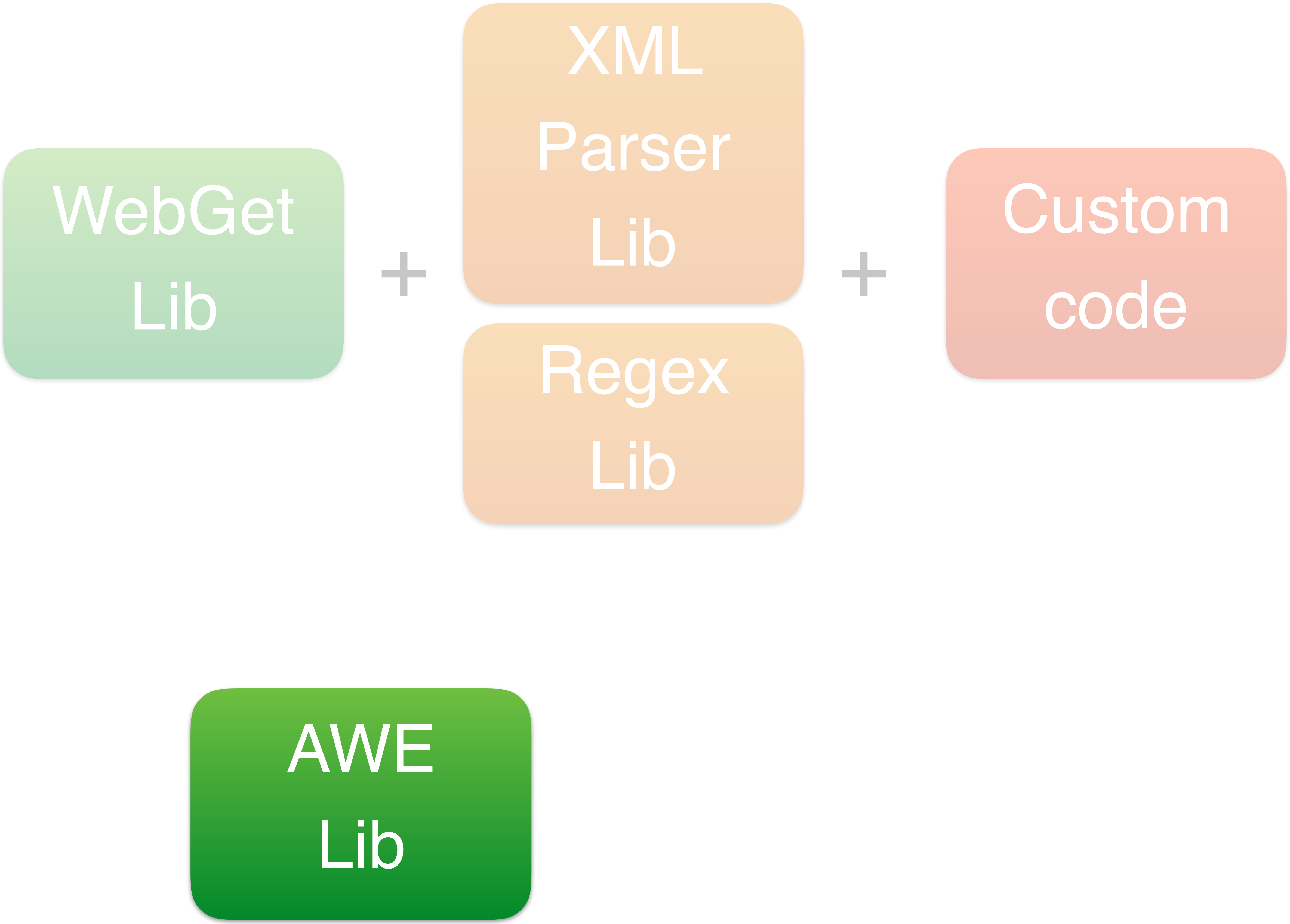
+

Web extraction, traditional solution vs AdaptiLib



WebGet Lib + XML Parser Lib / Regex Lib + Custom code

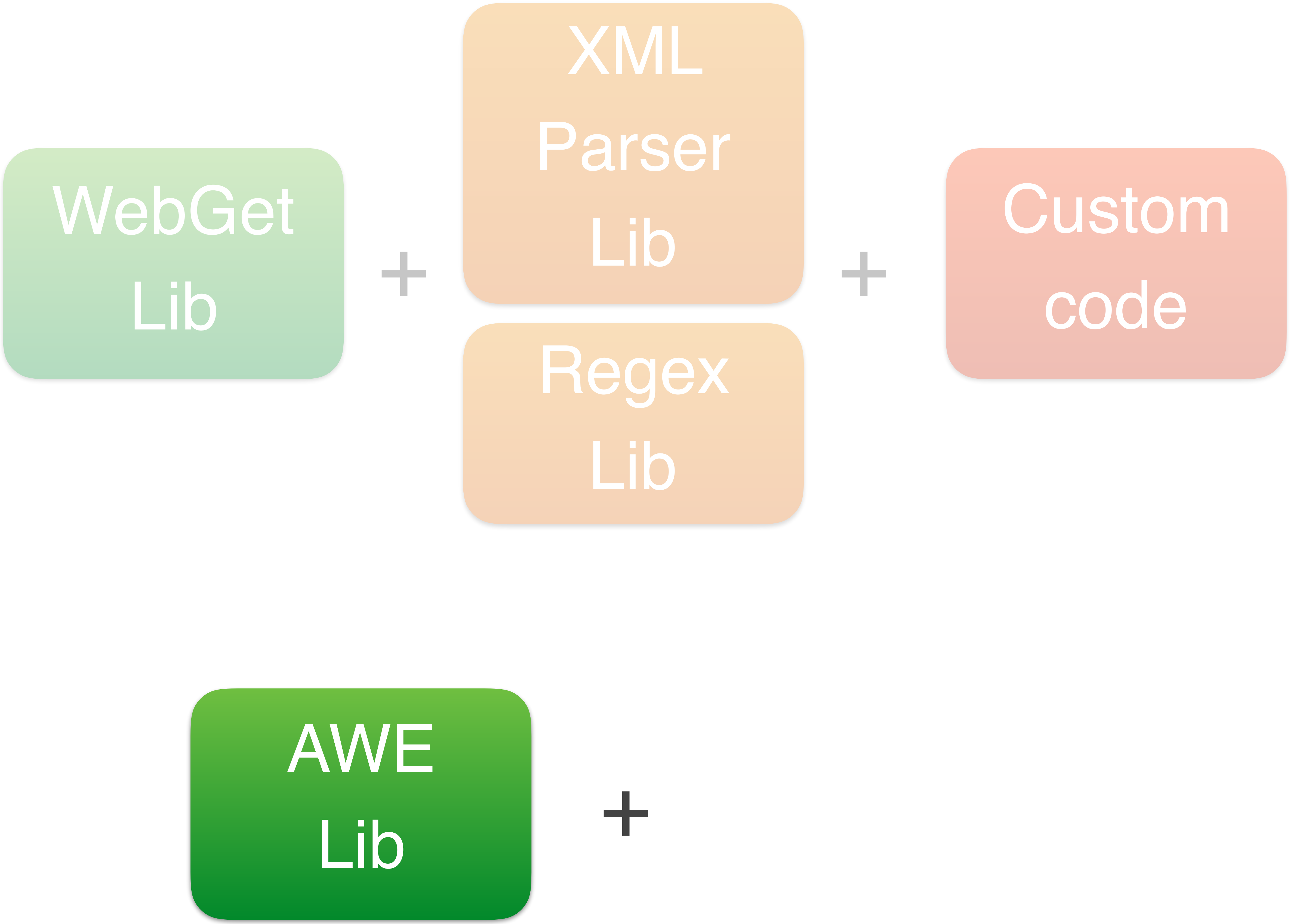
Web extraction, traditional solution vs AdaptiLib



WebGet Lib + XML Parser Lib / Regex Lib + Custom code

AWE Lib

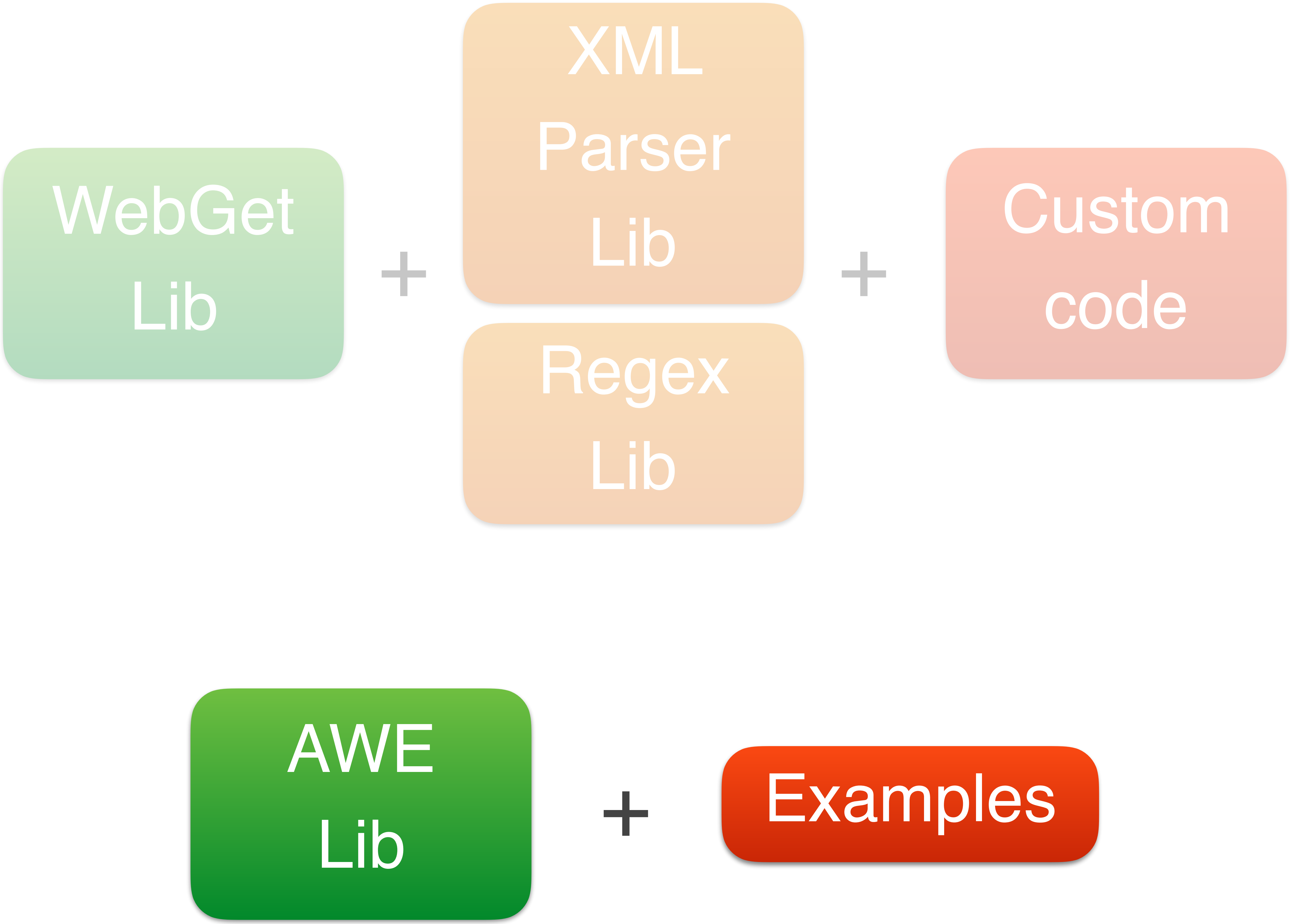
Web extraction, traditional solution vs AdaptiLib



WebGet Lib + XML Parser Lib / Regex Lib + Custom code

AWE Lib +

Web extraction, traditional solution vs AdaptiLib



WebGet Lib + XML Parser Lib / Regex Lib + Custom code

AWE Lib + Examples

Adaptive Libraries

Adaptive Libraries

- A normal library (lib):

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions
  - 4. and examples to understand API & how to put together

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions
  - 4. and examples to understand API & how to put together
- But only 1 above is directly useable without a human

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions
  - 4. and examples to understand API & how to put together
- But only 1 above is directly useable without a human
  - 2-4 requires a human to assemble solution based on text

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions
  - 4. and examples to understand API & how to put together
- But only 1 above is directly useable without a human
  - 2-4 requires a human to assemble solution based on text
- Adaptive libraries (AdaptiLibs):

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions
  - 4. and examples to understand API & how to put together
- But only 1 above is directly useable without a human
  - 2-4 requires a human to assemble solution based on text
- Adaptive libraries (AdaptiLibs):
  - 1. Still has basic “atoms” = functions to be called

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions
  - 4. and examples to understand API & how to put together
- But only 1 above is directly useable without a human
  - 2-4 requires a human to assemble solution based on text
- Adaptive libraries (AdaptiLibs):
  - 1. Still has basic “atoms” = functions to be called
  - (2a) But also executable examples that uses atoms to perform specific, named sequences

Adaptive Libraries

- A normal library (lib):
  - 1. has a number of functions that can be called
  - 2. to solve specific tasks
  - 3. has documentation to describe the functions
  - 4. and examples to understand API & how to put together
- But only 1 above is directly useable without a human
  - 2-4 requires a human to assemble solution based on text
- Adaptive libraries (AdaptiLibs):
  - 1. Still has basic “atoms” = functions to be called
  - (2a) But also executable examples that uses atoms to perform specific, named sequences
  - (2b) And allow fuzzy mapping of user needs to tasks

Example: Adaptive Web Extraction (AWE!) library, in practice

Example: Adaptive Web Extraction (AWE!) library, in practice

```
examples = [
(“scholar.google.se/citations?user=B3C4aY8AAAAJ&hl=en”,
{“name”: “V Basili”,
   “citations”: 33501,
   “h-index”: 82}),
(“scholar.google.se/citations?user=Zj897NoAAAAJ&hl=en”,
{“name”: “Lionel Briand”,
   “citations”: 21505,
   “h-index”: 69})]
```

Example: Adaptive Web Extraction (AWE!) library, in practice

```
examples = [
(“scholar.google.se/citations?user=B3C4aY8AAAAJ&hl=en”,
{“name”: “V Basili”,
   “citations”: 33501,
   “h-index”: 82}),
(“scholar.google.se/citations?user=Zj897NoAAAAJ&hl=en”,
{“name”: “Lionel Briand”,
   “citations”: 21505,
   “h-index”: 69})]

gscholar_ex = create_extractor(examples)
```

Example: Adaptive Web Extraction (AWE!) library, in practice

```
examples = [
("scholar.google.se/citations?user=B3C4aY8AAAAJ&hl=en",
{"name": "V Basili",
   "citations": 33501,
   "h-index": 82}),
("scholar.google.se/citations?user=Zj897NoAAAAJ&hl=en",
{"name": "Lionel Briand",
   "citations": 21505,
   "h-index": 69})]

gscholar_ex = create_extractor(examples)

extract(gscholar_ex, "scholar.google.se/citations?
user=CQDOm2gAAAAJ&hl=en")
```

Example: Adaptive Web Extraction (AWE!) library, in practice

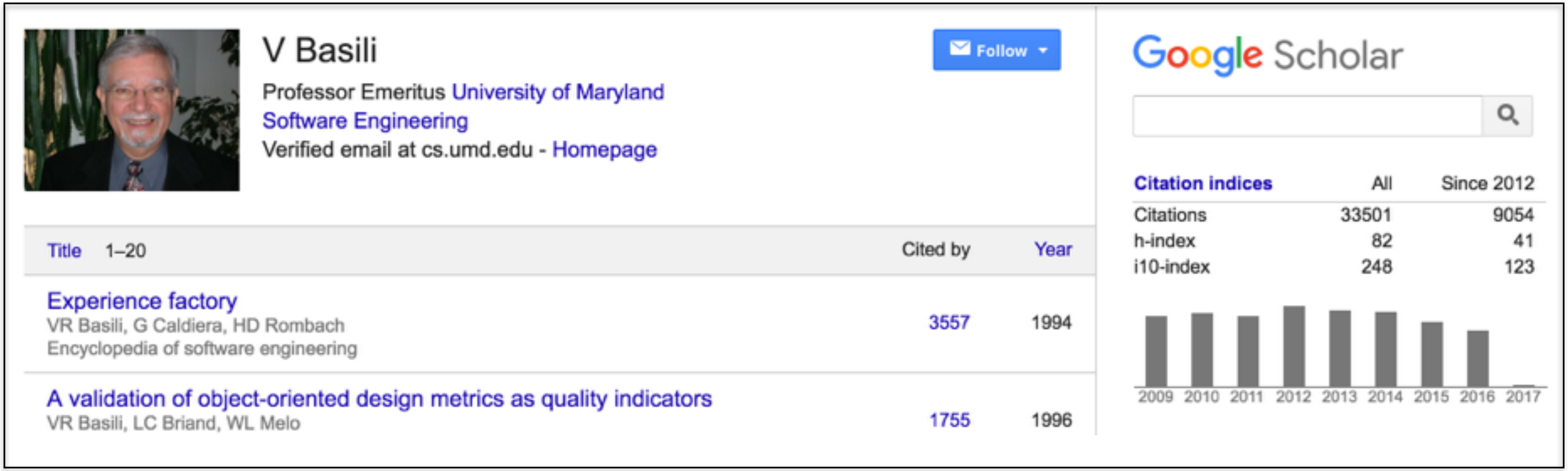
```
examples = [
(“scholar.google.se/citations?user=B3C4aY8AAAAJ&hl=en”,
{“name”: “V Basili”,
   “citations”: 33501,
   “h-index”: 82}),
(“scholar.google.se/citations?user=Zj897NoAAAAJ&hl=en”,
{“name”: “Lionel Briand”,
   “citations”: 21505,
   “h-index”: 69})]

gscholar_ex = create_extractor(examples)

extract(gscholar_ex, “scholar.google.se/citations?
user=CQDOm2gAAAAJ&hl=en”)

# returns:
# {“name”: “Barbara Ann Kitchenham”,
#  “citations”: 63,
#  “h-index”: 154})]
```

Big benefits with semantically similar task



V Basili

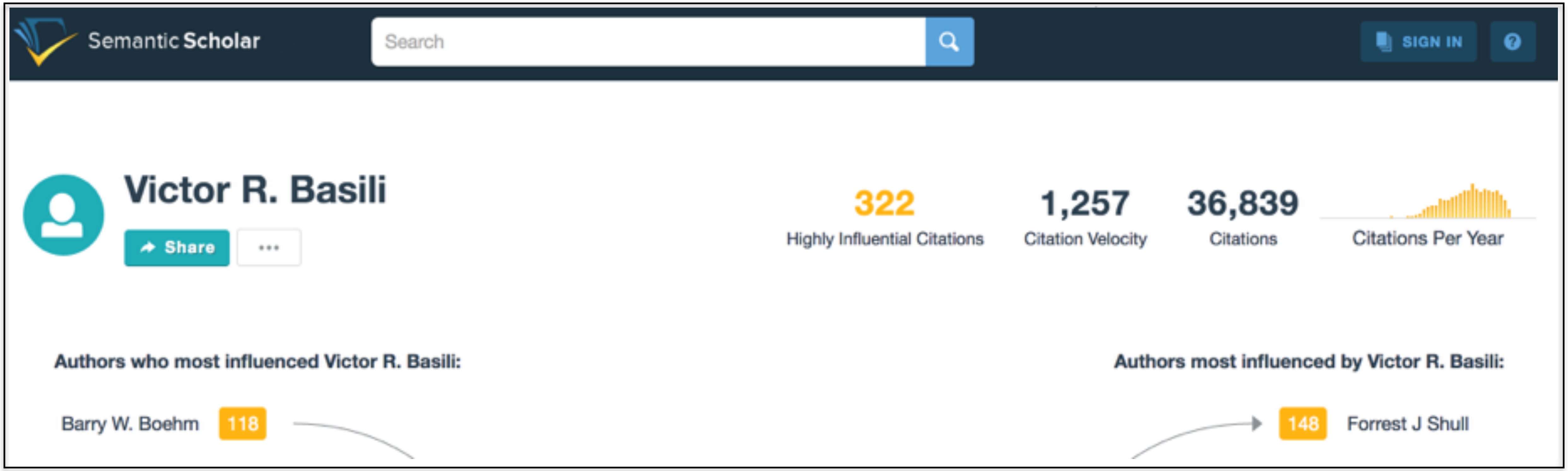
Professor Emeritus University of Maryland
Software Engineering
Verified email at cs.umd.edu - Homepage

Title 1–20	Cited by	Year
Experience factory VR Basili, G Caldiera, HD Rombach Encyclopedia of software engineering	3557	1994
A validation of object-oriented design metrics as quality indicators VR Basili, LC Briand, WL Melo	1755	1996

Citation indices	All	Since 2012
Citations	33501	9054
h-index	82	41
i10-index	248	123

```
{
   “name”: “V Basili”,
   “citations”: 33501,
   “h-index”: 82
}
```

Big benefits with semantically similar task



Victor R. Basili

322 Highly Influential Citations

1,257 Citation Velocity

36,839 Citations

Citations Per Year

Authors who most influenced Victor R. Basili:

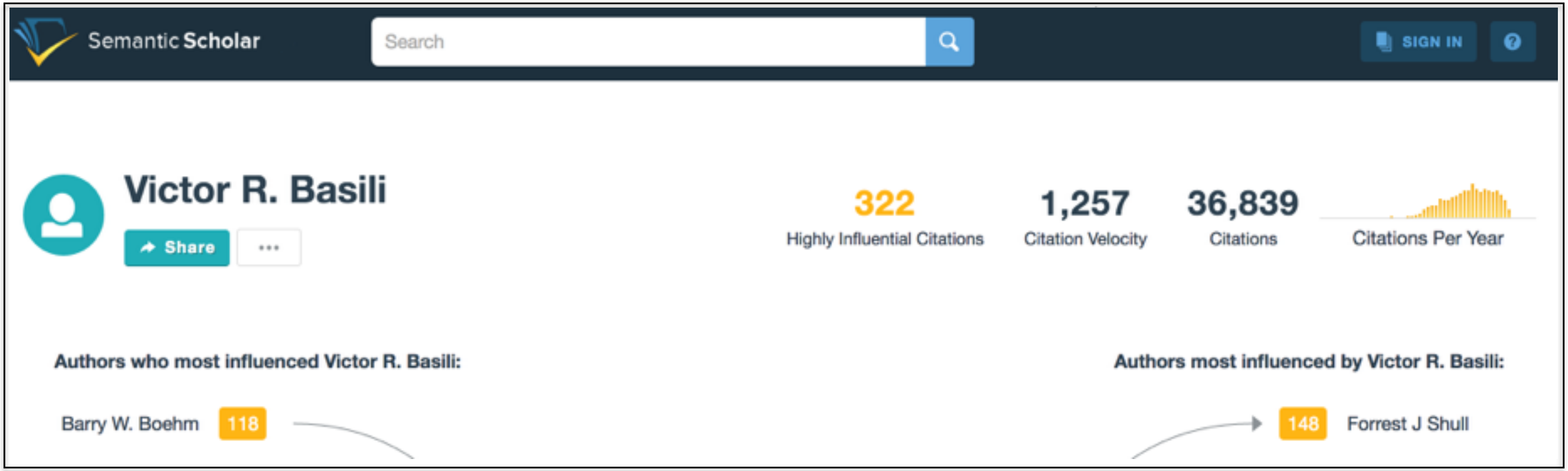
Barry W. Boehm 118

Authors most influenced by Victor R. Basili:

148 Forrest J Shull

```
{
   "name": "V Basili",
   "citations": 33501,
   "h-index": 82
}
```

Big benefits with semantically similar task



Victor R. Basili

322
Highly Influential Citations

1,257
Citation Velocity

36,839
Citations

Citations Per Year

Authors who most influenced Victor R. Basili:

Barry W. Boehm 118

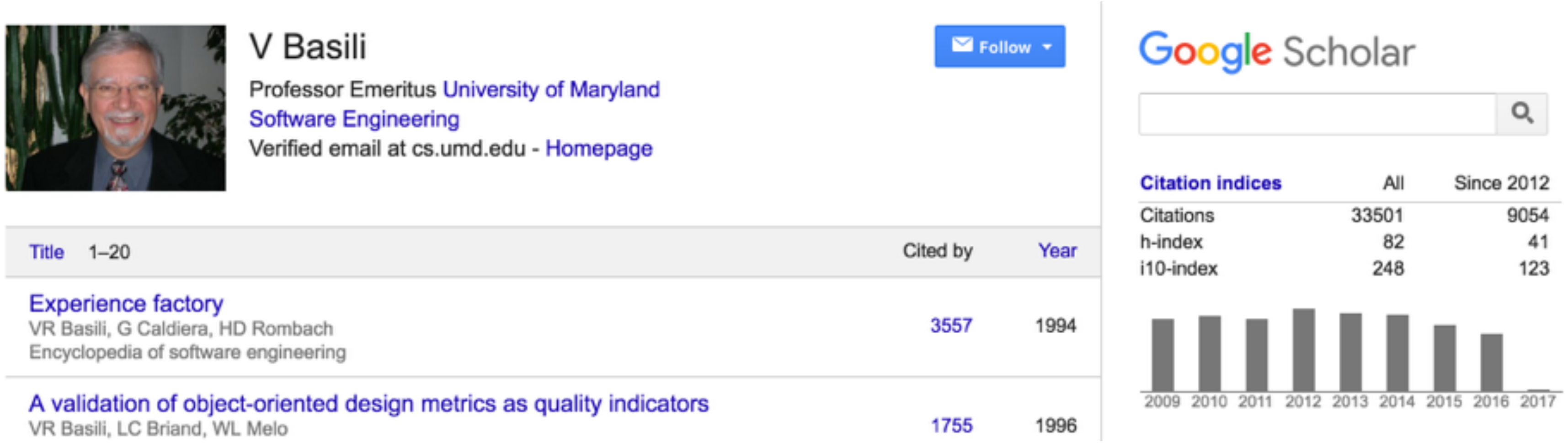
Authors most influenced by Victor R. Basili:

148 Forrest J Shull

```
{
    “name”: “Victor R. Basili”,
    “citations”: 36839,
    “influential”: 322
}
```

Only change 2 I/O examples & re-adapt!

GI would not help: Only semantic, not syntactic similarity



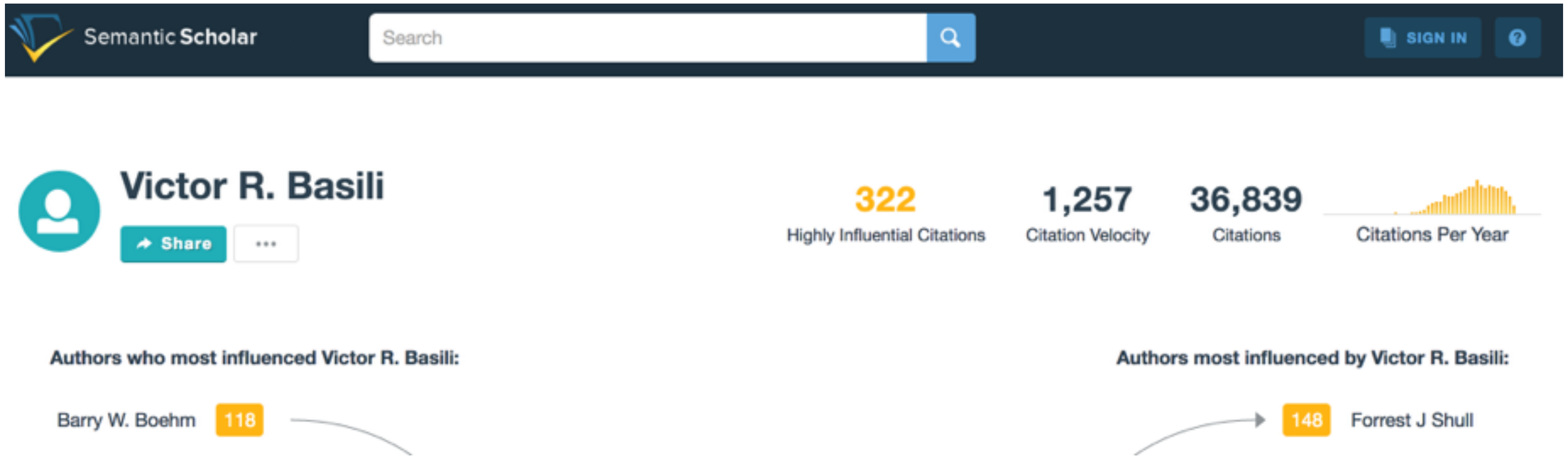
V Basili

Professor Emeritus University of Maryland
Software Engineering
Verified email at cs.umd.edu - Homepage

Title 1–20	Cited by	Year
Experience factory VR Basili, G Caldiera, HD Rombach Encyclopedia of software engineering	3557	1994
A validation of object-oriented design metrics as quality indicators VR Basili, LC Briand, WL Melo	1755	1996

Citation indices	All	Since 2012
Citations	33501	9054
h-index	82	41
i10-index	248	123

```
"...>Citations</a></td><td class="gsc_rsb_std">33501</
td><td class="gsc_rsb_std">9054</td>..."
```



Victor R. Basili

322	1,257	36,839	
Highly Influential Citations	Citation Velocity	Citations	Citations Per Year

Authors who most influenced Victor R. Basili:

Barry W. Boehm 118

Authors most influenced by Victor R. Basili:

148 Forrest J Shull

```
"...:{"hIndex":51,"estimatedTotalCitationCount":{"min":
31675,"value":36839,"max":42905,..."
```

Design Rules for AdaptiLibs (so far…)

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase
  - Basic data access: get_url

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase
  - Basic data access: get_url
  - Matching: matchregexp, matchregexp_ignorecase

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase
  - Basic data access: get_url
  - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
    - Type conversion operations: parseToInt, parseToFloat
    - Data transformation: uppercase, lowercase, leadingcase
    - Basic data access: get_url
    - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible
    - Extend with atoms, and possibly (complex) atom seq.

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
    - Type conversion operations: parseToInt, parseToFloat
    - Data transformation: uppercase, lowercase, leadingcase
    - Basic data access: get_url
    - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible
    - Extend with atoms, and possibly (complex) atom seq.
- Feldt’s Law for Designing Lib incl. Search, consider in order:

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase
  - Basic data access: get_url
  - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible
  - Extend with atoms, and possibly (complex) atom seq.
- Feldt’s Law for Designing Lib incl. Search, consider in order:
  - 1. Deterministic / Exact (fastest, most efficient)

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase
  - Basic data access: get_url
  - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible
  - Extend with atoms, and possibly (complex) atom seq.
- Feldt’s Law for Designing Lib incl. Search, consider in order:
  - 1. Deterministic / Exact (fastest, most efficient)
  - 2. Heuristics / Approximations (order by applicability)

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase
  - Basic data access: get_url
  - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible
  - Extend with atoms, and possibly (complex) atom seq.
- Feldt’s Law for Designing Lib incl. Search, consider in order:
  - 1. Deterministic / Exact (fastest, most efficient)
  - 2. Heuristics / Approximations (order by applicability)
  - 3. Focused Search (part of solution only, then aggregate)

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
  - Type conversion operations: parseToInt, parseToFloat
  - Data transformation: uppercase, lowercase, leadingcase
  - Basic data access: get_url
  - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible
  - Extend with atoms, and possibly (complex) atom seq.
- Feldt’s Law for Designing Lib incl. Search, consider in order:
  - 1. Deterministic / Exact (fastest, most efficient)
  - 2. Heuristics / Approximations (order by applicability)
  - 3. Focused Search (part of solution only, then aggregate)
  - 4. Interact / Ask Developer (in adapt step)

Design Rules for AdaptiLibs (so far…)

- Start by defining basic “atomic” operations
    - Type conversion operations: parseToInt, parseToFloat
    - Data transformation: uppercase, lowercase, leadingcase
    - Basic data access: get_url
    - Matching: matchregexp, matchregexp_ignorecase
- Go through concrete task from example & note how a human solves it in as atomic steps as possible
    - Extend with atoms, and possibly (complex) atom seq.
- Feldt’s Law for Designing Lib incl. Search, consider in order:
    - 1. Deterministic / Exact (fastest, most efficient)
    - 2. Heuristics / Approximations (order by applicability)
    - 3. Focused Search (part of solution only, then aggregate)
    - 4. Interact / Ask Developer (in adapt step)
    - 5. Full/free search (search from atoms & up, warn dev)

Conclusions

Conclusions

- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming

Conclusions

- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming
  - Compared to other SBSE, GI comes closer to AP

Conclusions

- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming
    - Compared to other SBSE, GI comes closer to AP
- As techniques and processing power increase we will see more practical AP

Conclusions

- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming
  - Compared to other SBSE, GI comes closer to AP
- As techniques and processing power increase we will see more practical AP
  - But semantic similarity does not imply syntactic similarity => less opportunity for detailed code reuse

Conclusions

- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming
  - Compared to other SBSE, GI comes closer to AP
- As techniques and processing power increase we will see more practical AP
  - But semantic similarity does not imply syntactic similarity => less opportunity for detailed code reuse
- But we can also deliver practical AP now by hybridising it with human intelligence and guidance

Conclusions

- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming
  - Compared to other SBSE, GI comes closer to AP
- As techniques and processing power increase we will see more practical AP
  - But semantic similarity does not imply syntactic similarity => less opportunity for detailed code reuse
- But we can also deliver practical AP now by hybridising it with human intelligence and guidance
- We are developing AdaptiLibs, general libraries that adapt to I/O examples of users/developers

Conclusions

- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming
  - Compared to other SBSE, GI comes closer to AP
- As techniques and processing power increase we will see more practical AP
  - But semantic similarity does not imply syntactic similarity => less opportunity for detailed code reuse
- But we can also deliver practical AP now by hybridising it with human intelligence and guidance
- We are developing AdaptiLibs, general libraries that adapt to I/O examples of users/developers
  - Combines task-driven design & experience of humans

Conclusions

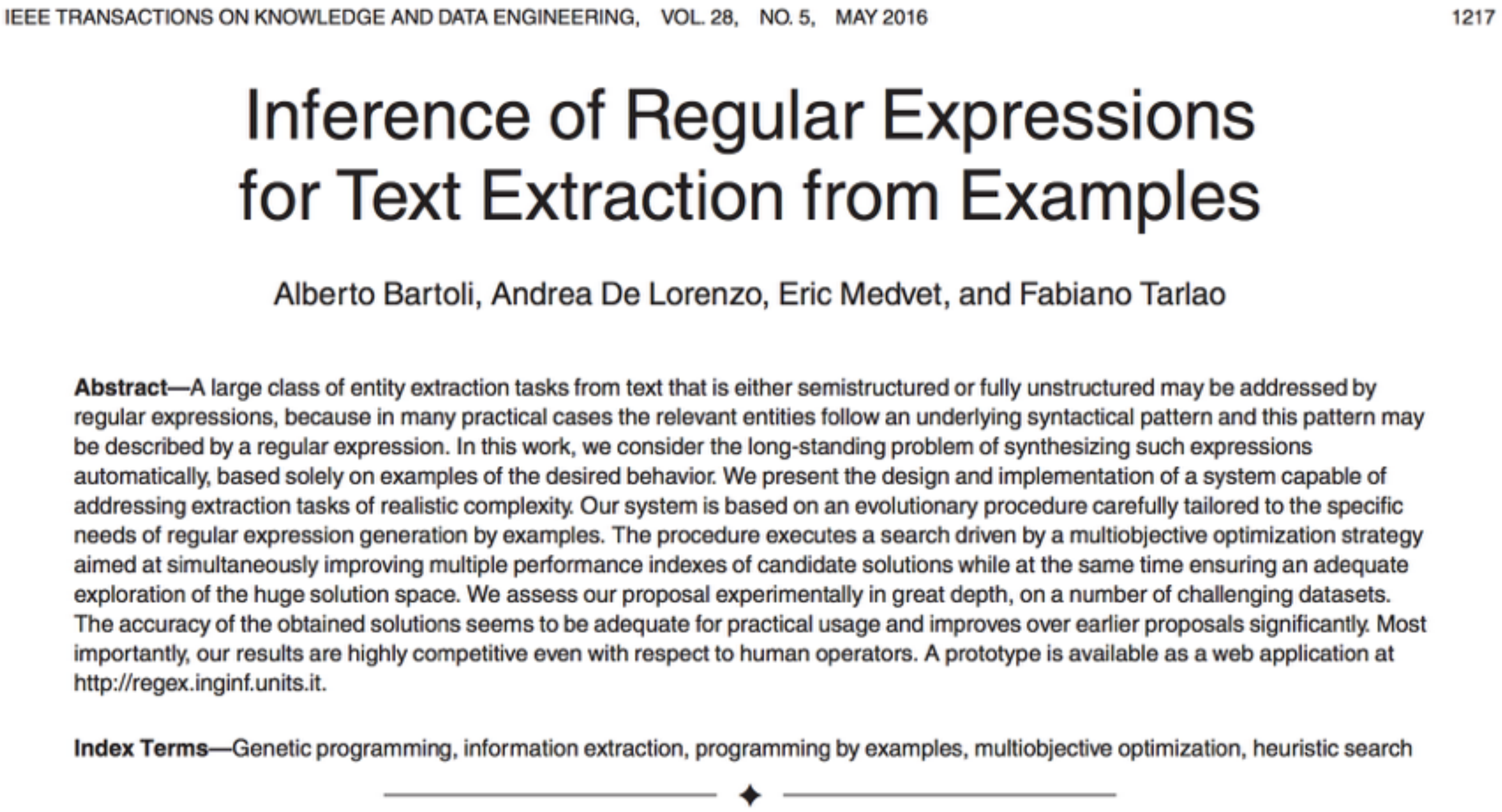
- Despite many promises of GP & SBSE it has under delivered on practical Automated Programming
    - Compared to other SBSE, GI comes closer to AP
- As techniques and processing power increase we will see more practical AP
    - But semantic similarity does not imply syntactic similarity => less opportunity for detailed code reuse
- But we can also deliver practical AP now by hybridising it with human intelligence and guidance
- We are developing AdaptiLibs, general libraries that adapt to I/O examples of users/developers
    - Combines task-driven design & experience of humans
    - with brute force and flexibility of search, only wh. needed

Thank you!

robert.feldt@chalmers.se

@drfeldt

But what about Bartoli et al?!



IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 28, NO. 5, MAY 2016 1217

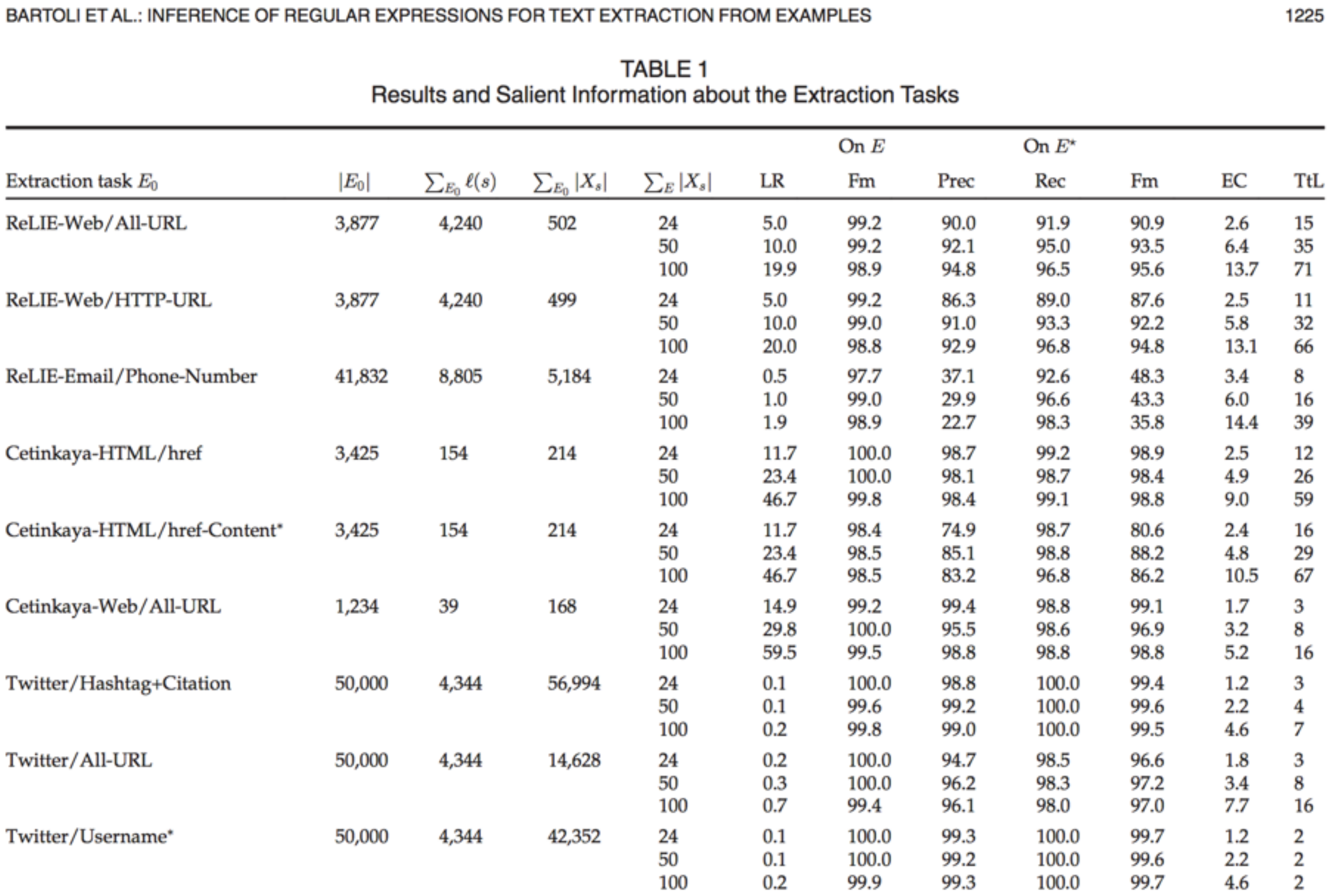
Inference of Regular Expressions for Text Extraction from Examples

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao

Abstract—A large class of entity extraction tasks from text that is either semistructured or fully unstructured may be addressed by regular expressions, because in many practical cases the relevant entities follow an underlying syntactical pattern and this pattern may be described by a regular expression. In this work, we consider the long-standing problem of synthesizing such expressions automatically, based solely on examples of the desired behavior. We present the design and implementation of a system capable of addressing extraction tasks of realistic complexity. Our system is based on an evolutionary procedure carefully tailored to the specific needs of regular expression generation by examples. The procedure executes a search driven by a multiobjective optimization strategy aimed at simultaneously improving multiple performance indexes of candidate solutions while at the same time ensuring an adequate exploration of the huge solution space. We assess our proposal experimentally in great depth, on a number of challenging datasets. The accuracy of the obtained solutions seems to be adequate for practical usage and improves over earlier proposals significantly. Most importantly, our results are highly competitive even with respect to human operators. A prototype is available as a web application at http://regex.inginf.units.it.

Index Terms—Genetic programming, information extraction, programming by examples, multiobjective optimization, heuristic search

But what about Bartoli et al?!



BARTOLI ET AL.: INFERENCE OF REGULAR EXPRESSIONS FOR TEXT EXTRACTION FROM EXAMPLES 1225

TABLE 1
Results and Salient Information about the Extraction Tasks

						On E	On E^*				
Extraction task E_0	$\lvert E_0 \rvert$	$\sum_{E_0} \ell(s)$	$\sum_{E_0} \lvert X_s \rvert$	$\sum_{E} \lvert X_s \rvert$	LR	Fm	Prec	Rec	Fm	EC	TtL
ReLIE-Web/All-URL	3,877	4,240	502	24	5.0	99.2	90.0	91.9	90.9	2.6	15
				50	10.0	99.2	92.1	95.0	93.5	6.4	35
				100	19.9	98.9	94.8	96.5	95.6	13.7	71
ReLIE-Web/HTTP-URL	3,877	4,240	499	24	5.0	99.2	86.3	89.0	87.6	2.5	11
				50	10.0	99.0	91.0	93.3	92.2	5.8	32
				100	20.0	98.8	92.9	96.8	94.8	13.1	66
ReLIE-Email/Phone-Number	41,832	8,805	5,184	24	0.5	97.7	37.1	92.6	48.3	3.4	8
				50	1.0	99.0	29.9	96.6	43.3	6.0	16
				100	1.9	98.9	22.7	98.3	35.8	14.4	39
Cetinkaya-HTML/href	3,425	154	214	24	11.7	100.0	98.7	99.2	98.9	2.5	12
				50	23.4	100.0	98.1	98.7	98.4	4.9	26
				100	46.7	99.8	98.4	99.1	98.8	9.0	59
Cetinkaya-HTML/href-Content*	3,425	154	214	24	11.7	98.4	74.9	98.7	80.6	2.4	16
				50	23.4	98.5	85.1	98.8	88.2	4.8	29
				100	46.7	98.5	83.2	96.8	86.2	10.5	67
Cetinkaya-Web/All-URL	1,234	39	168	24	14.9	99.2	99.4	98.8	99.1	1.7	3
				50	29.8	100.0	95.5	98.6	96.9	3.2	8
				100	59.5	99.5	98.8	98.8	98.8	5.2	16
Twitter/Hashtag+Citation	50,000	4,344	56,994	24	0.1	100.0	98.8	100.0	99.4	1.2	3
				50	0.1	99.6	99.2	100.0	99.6	2.2	4
				100	0.2	99.8	99.0	100.0	99.5	4.6	7
Twitter/All-URL	50,000	4,344	14,628	24	0.2	100.0	94.7	98.5	96.6	1.8	3
				50	0.3	100.0	96.2	98.3	97.2	3.4	8
				100	0.7	99.4	96.1	98.0	97.0	7.7	16
Twitter/Username*	50,000	4,344	42,352	24	0.1	100.0	99.3	100.0	99.7	1.2	2
				50	0.1	100.0	99.2	100.0	99.6	2.2	2
				100	0.2	99.9	99.3	100.0	99.7	4.6	2