Mining Specifications from Documentation Using a Crowd

*Peng Sun
^Ivan Beschastnikh
*Chris Brown
*Kathryn Stolee

* NC State University
^ University of British Columbia

+

Overview Package Class Use Tree Deprecated Index Help
Java™ Platform Standard Ed. 6

java.util
Interface List<E>

All Superinterfaces:
Collection<E>, Iterable<E>

All Known Implementing Classes:
AbstractList, AbstractSequentialList, ArrayList, AttributeList, CopyOnWriteArrayList, LinkedList, RoleList, RoleUnresolvedList, Stack, Vector

```
public interface List<E>
extends Collection<E>
```

An ordered collection (also known as a *sequence*). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements e1 and e2 such that e1.equals(e2), and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

The List interface places additional stipulations, beyond those specified in the Collection interface, on the contracts of the iterator, add, remove, equals, and hashCode methods. Declarations for other inherited methods are also included here for convenience.

The List interface provides four methods for positional (indexed) access to list elements. Lists (like Java arrays) are zero based. Note that these operations may execute in time proportional to the index value for some implementations (the LinkedList class, [illegible]

+

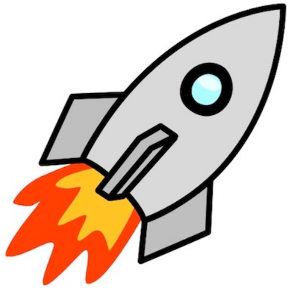
Mining Specifications from Documentation Using a Crowd

*Peng Sun *Chris Brown
^*Ivan Beschastnikh* *Kathryn Stolee

* NC State University
^ University of British Columbia

Software Specifications

Software systems and libraries usually lack up-to-date formal specifications.



Rapid Software Evolution

Formal specifications are non-trivial to write down

Software Specifications

Lack of Formal Specifications

↓

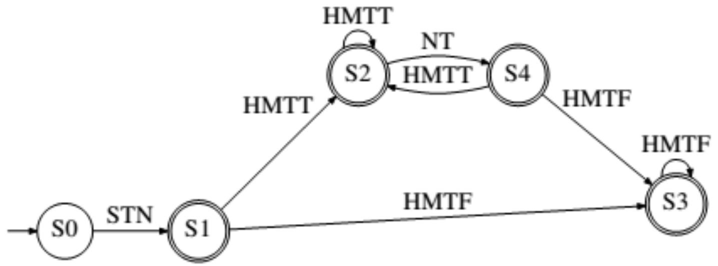
Maintainability & Reliability Challenges

- Reduced code comprehension
- Implicit assumptions may cause bugs
- Difficult to identify regressions

Software Specification Mining

Software Specifications Mining

- Many existing specification mining algorithms
  - Most automatically infer specs from *execution traces*



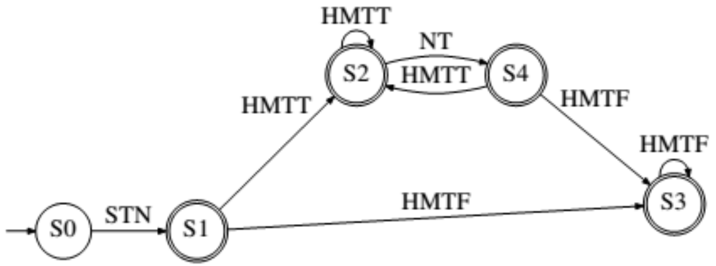
Finite State Automata (FSA)

Examples: k-tail, CONTRACTOR++, SEKT, TEMI, Synoptic,…

TSE 1972,
ICSE 2006,
ASE 2009,
FSE 2011,
FSE 2014,
ICSE 2014,
TSE 2015,
ASE 2015,
…

Software Specifications Mining

- Many existing specification mining algorithms
  - Most automatically infer specs from *execution traces*


$$\begin{tikzcd}
 & & S2 \arrow[loop above, "HMTT"] \arrow[r, bend left, "NT"] & S4 \arrow[l, bend left, "HMTT"] \arrow[dr, "HMTF"] & \\
\arrow[r] & S0 \arrow[r, "STN"] & S1 \arrow[u, "HMTT"] \arrow[rr, "HMTF"] & & S3 \arrow[loop above, "HMTF"]
\end{tikzcd}$$

Finite State Automata (FSA)

Examples: k-tail, CONTRACTOR++, SEKT, TEMI, Synoptic,…

TSE 1972,
ICSE 2006,
ASE 2009,
FSE 2011,
FSE 2014,
ICSE 2014,
TSE 2015,
ASE 2015,
…

But, automation is a dimension

Prior to 1990s

Entirely Manual

Formal methods experts

But, automation is a dimension

Prior to 1990s

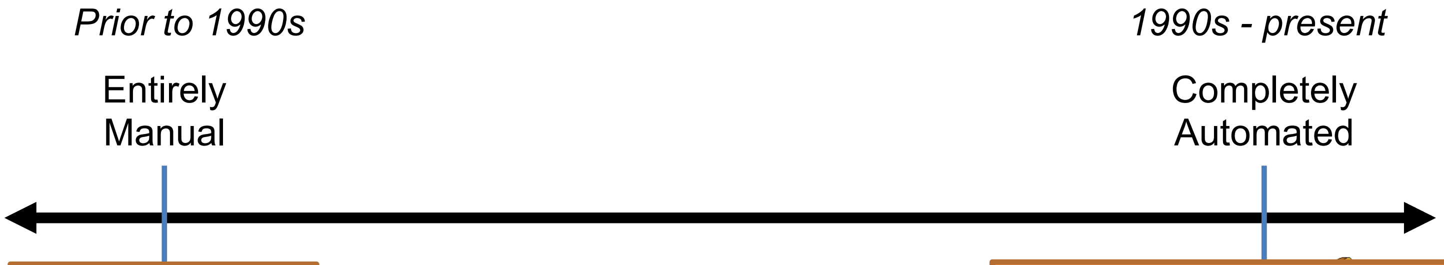
Entirely
Manual

1990s - present

Completely
Automated

Formal methods experts

But, automation is a dimension



- Expensive
- Not scalable

Formal methods experts

- False positives
- Requires artifact diversity
- Requires accurate artifacts

Our contribution: <u>crowd</u> spec mining from docs

Prior to 1990s

Entirely Manual

- Expensive
- Not scalable

Formal methods experts

SANER 2019

Crowd Mining

1990s - present

Completely Automated

- False positives
- Requires artifact diversity
- Requires accurate artifacts

Prior to 1990s — Entirely Manual

SANER 2019 — **Crowd Mining**

1990s - present — Completely Automated

Formal methods experts

RQ1: Can crowd do as well as experts?

RQ2: Can crowd improve, or replace, existing spec miners?

Crowd-sourcing in SE (not a new idea)

- Crowd is effective at a variety of SE tasks
  - Testing [1]
  - Evaluating code smells [2]
  - Program synthesis [3]
  - Building software [4]

[1] Dolstra et al. *Crowdsourcing GUI tests.* ICST 2013.
[2] Stolee et al. *Exploring the use of crowdsourcing to support empirical studies in software engineering.* ESEM 2010.
[3] Cochran et al. *Program boosting: Program synthesis via crowd-sourcing.* SIGPLAN Not. Vol. 50 No. 1. L2015
[4] LaToza et al. *Microtask programming: Building software with a crowd.* UIST 2014.

Crowd-sourcing in SE (not a new idea)

- Crowd is effective at a variety of SE tasks
- Prior work on crowd mining HW specs [5]. We differ:
  - Use docs instead of traces, SW specs not HW
  - We use standard quality controls, not gamification
  - We improve spec miners/compare to experts

[1] Dolstra et al. *Crowdsourcing GUI tests.* ICST 2013.
[2] Stolee et al. *Exploring the use of crowdsourcing to support empirical studies in software engineering.* ESEM 2010.
[3] Cochran et al. *Program boosting: Program synthesis via crowd-sourcing.* SIGPLAN Not. Vol. 50 No. 1. 2015
[4] LaToza et al. *Microtask programming: Building software with a crowd.* UIST 2014.
[5] Li et al. *Crowdmine: Towards crowdsourced human-assisted verification. DAC 2012.*

Crowd-sourcing spec mining [CrowdSpec]

Design questions to answer:

- What kind of spec to mine?
- What resource to mine specs from?
- How to solicit contributions from the crowd?
- How to combine crowd responses?

Crowd-sourcing spec mining [CrowdSpec]

Design question/answers:

- Type of spec? **Temporal APIs**
- What resource? **Documentation**
- How to solicit? **MTurk microtasks**
- Combining responses? **Voting**

Crowd-sourcing spec mining [CrowdSpec]

Good for humans, if simple

Design question/answers:

Aligns with prior work (can compare)

Notoriously difficult [1]; crowd could help?

- Type of spec? **Temporal APIs**
- What resource? **Documentation**
- How to solicit? **MTurk microtasks**
- Combining responses? **Voting**

[1] Legunsen et al. *How good are the specs? a study of the bug-finding effectiveness of existing java api specifications.* ASE 2016.

Crowd-sourcing spec mining [CrowdSpec]

Design question/answers:

- Type of spec? **Temporal APIs**
- What resource? **Documentation**
- How to solicit? **MTurk microtasks**
- Combining responses? **Voting**

Great for humans (beats traces!)

Very few existing spec miners [1]

Good temporal NLP is hard

[1] Pandita et al. *ICON: Inferring temporal constraints from natural language API descriptions.* ICSME 2016.

Crowd-sourcing spec mining [CrowdSpec]

Design question/answers:

- Type of spec? **Temporal APIs**
- What resource? **Documentation**
- How to solicit? **MTurk microtasks**
- Combining responses? **Voting**

Existing platform with critical mass

Well-defined econ model: pay per HIT (**H**uman **I**ntelligence **T**ask)

Crowd-sourcing spec mining [CrowdSpec]

Design question/answers:

- Type of spec? **Temporal APIs**
- What resource? **Documentation**
- How to solicit? **MTurk microtasks**
- Combining responses? **Voting**

Lots of flexibility

Implements reliability

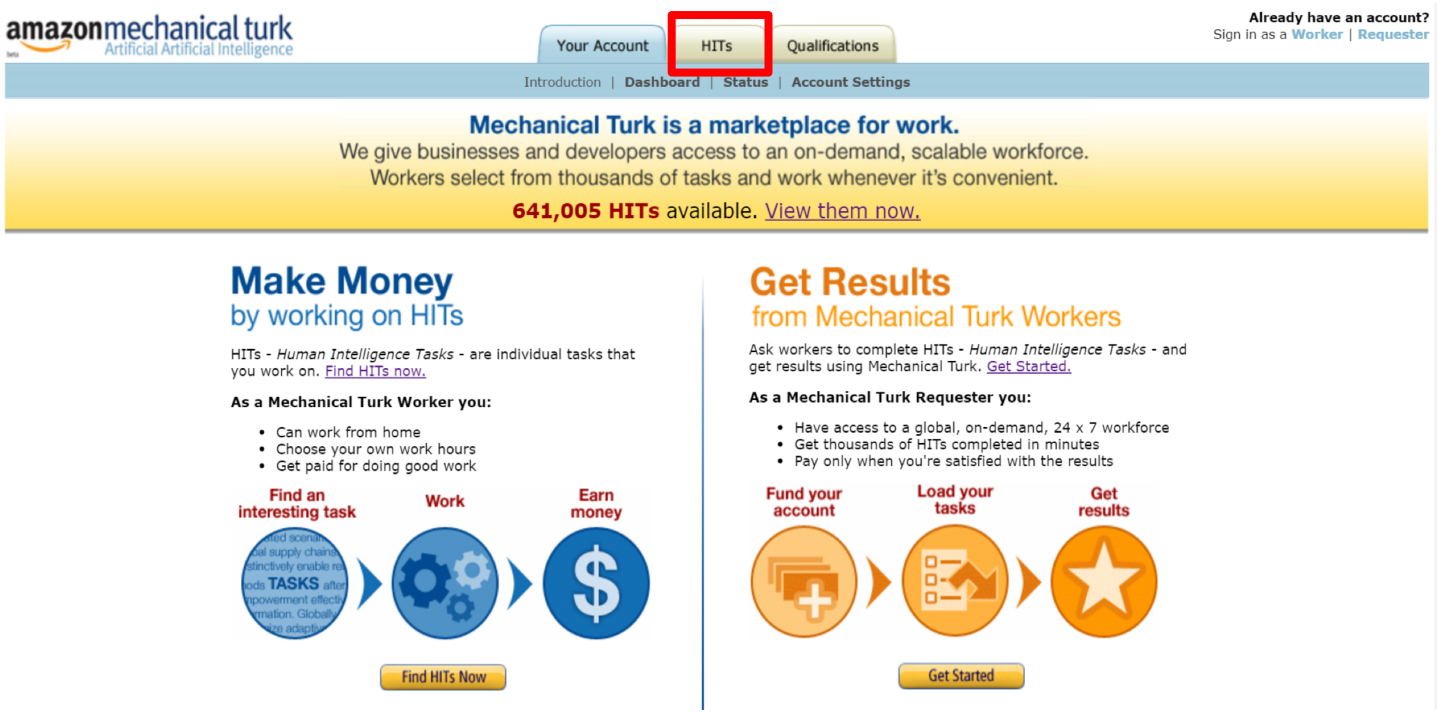
CrowdSpec contributions

- CrowdSpec + SpecForge [1] can perform as well as voting experts: powerful hybrid spec mining alternatives

- Qualitative analysis of where crowd made mistakes

[1] T-D. B. et al. *Synergizing specification miners through model fissions and fusions.* ASE 2015.

Approach overview



Already have an account?
Sign in as a Worker | Requester

Mechanical Turk is a marketplace for work.
We give businesses and developers access to an on-demand, scalable workforce.
Workers select from thousands of tasks and work whenever it's convenient.
641,005 HITs available. View them now.

Make Money
by working on HITs

HITs - *Human Intelligence Tasks* - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:

- Can work from home
- Choose your own work hours
- Get paid for doing good work

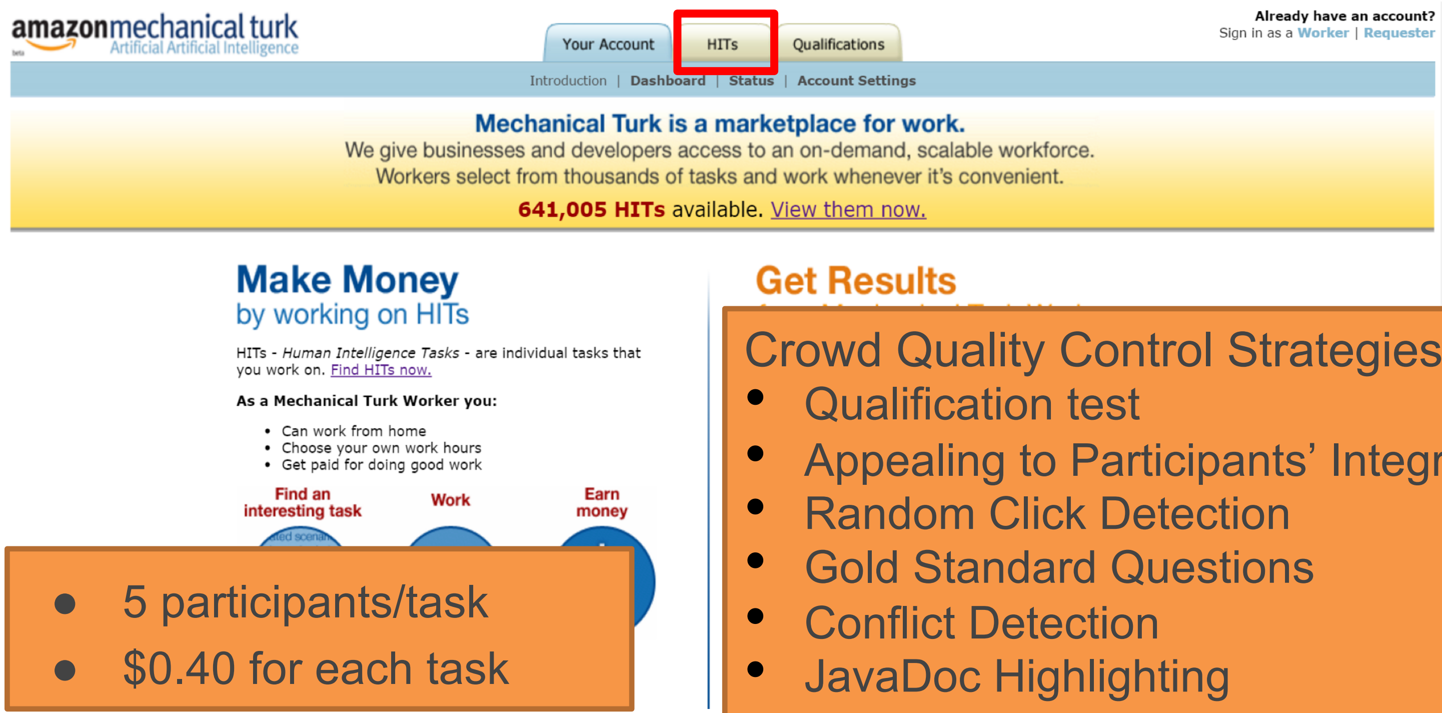
Get Results
from Mechanical Turk Workers

Ask workers to complete HITs - *Human Intelligence Tasks* - and get results using Mechanical Turk. Get Started.

As a Mechanical Turk Requester you:

- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

Approach overview


amazonmechanical turk
Artificial Artificial Intelligence

Your Account | HITs | Qualifications

Already have an account?
Sign in as a Worker | Requester

Introduction | Dashboard | Status | Account Settings

Mechanical Turk is a marketplace for work.
We give businesses and developers access to an on-demand, scalable workforce.
Workers select from thousands of tasks and work whenever it's convenient.
641,005 HITs available. View them now.

Make Money
by working on HITs

HITs - *Human Intelligence Tasks* - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:

- Can work from home
- Choose your own work hours
- Get paid for doing good work

Get Results

- 5 participants/task
- $0.40 for each task

Crowd Quality Control Strategies:

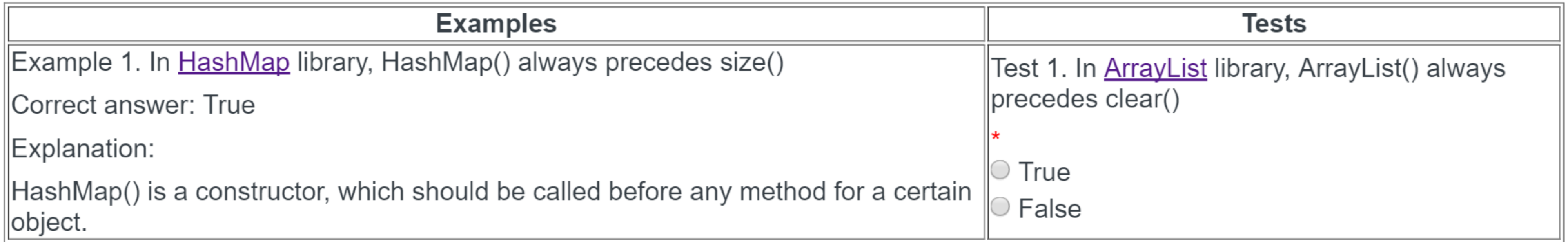
- Qualification test
- Appealing to Participants' Integrity
- Random Click Detection
- Gold Standard Questions
- Conflict Detection
- JavaDoc Highlighting

The crowd must be controlled

“Where there is power, there is resistance.” -- Foucault

Qualification test:

One question from the Qualification Test.



Examples	Tests
Example 1. In HashMap library, HashMap() always precedes size() Correct answer: True Explanation: HashMap() is a constructor, which should be called before any method for a certain object.	Test 1. In ArrayList library, ArrayList() always precedes clear() * ○ True ○ False

Study Design

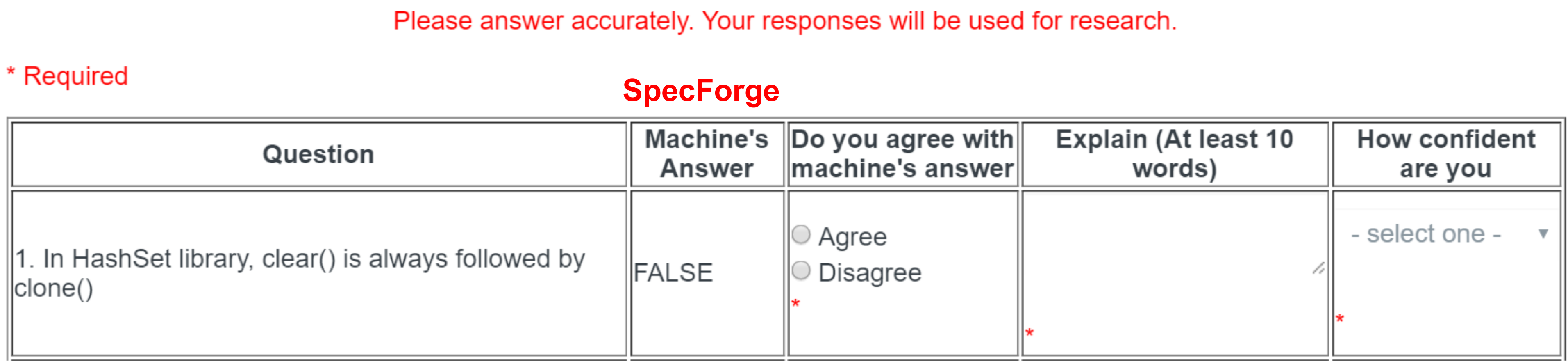
Task Design:

The API document link for library "HashSet"	java.util.HashSet
clear()	public void clear() Removes all of the elements from this set. The set will be empty after this call returns. Specified by: clear in interface Collection Specified by: clear in interface Set Overrides: clear in class AbstractCollection
clone()	public Object clone() Returns a shallow copy of this HashSet instance: the elements themselves are not cloned. Overrides: clone in class Object Returns: a shallow copy of this set See Also: Cloneable

Study Design

Task Design:

HIT with one temporal property (Always Followed By) for clear() and clone():



Please answer accurately. Your responses will be used for research.

* Required

SpecForge

Question	Machine's Answer	Do you agree with machine's answer	Explain (At least 10 words)	How confident are you
1. In HashSet library, clear() is always followed by clone()	FALSE	○ Agree ○ Disagree *	*	- select one - ▾ *

Temporal Constraint Types

- AF(a,b) : a is always followed by b

a b a b ✓	a b b a ⊗
c b b b ✓	c a a a ⊗

- NF(a,b): a is never followed by b

b b a a ✓	a b b a ⊗
a c a a ✓	c b a b ⊗

- AP(b,a): b always precedes a

b b a a ✓	a b b b ⊗
c b b b ✓	c a a b ⊗

Temporal Constraint Types

- AF(a,b) : *a* is always followed by *b*

a b a b ✓	a b b a ⊗
c b b b ✓	c a a a ⊗

- NF(a,b): *a* is never followed by *b*

b b a a ✓	a b b a ⊗
a c a a ✓	c b a b ⊗

- AP(b,a): *b* always precedes *a*

b b a a ✓	a b b b ⊗
c b b b ✓	c a a b ⊗

Temporal Constraint Types

- AF(a,b): *a* is always followed by *b*

a b a b ✓	a b b a ⊗
c b b b ✓	c a a a ⊗

- NF(a,b): *a* is never followed by *b*

b b a a ✓	a b b a ⊗
a c a a ✓	c b a b ⊗

- AP(b,a): *b* always precedes *a*

b b a a ✓	a b b b ⊗
c b b b ✓	c a a b ⊗

The Immediate Temporal Constraints

- AIF(a,b): a is always *immediately* followed by b
- NIF(a,b): a is never *immediately* followed by b
- AIP(a,b): a always *immediately* precedes b

[1] Dwyer et al. *Patterns in Property Specifications for Finite-state Verification*, ICSE 1999

[2] Yang et al. *Perracotta: Mining temporal API rules from imperfect traces.* ICSE 2006.

AIF, NIF, and AIP are extensions of AF, NF, and AP

Temporal specification

True property:

A program that uses the API and does not follow the property may **trigger a Java exception**, or a violation of the property is **impossible in the Java language**.

Examples: *HashSet() always precedes size()*; ✓ *clear() is always followed by size().* ✗

Evaluation: ground truth specs

- Three paper authors manually labeled property instances
- Targeted 3 Java APIs
  - HashSet
  - StringTokenizer
  - StackAr

		Inter-rater Kappa	
API	**Instances**	**Agreement**	**% True**
`HashSet`	1,014	0.82	6% (56)
`StringTokenizer`	384	0.76	9% (35)
`StackAr`	600	0.76	7% (43)

CrowdSpec v. SpecForge

Study	**Accuracy**	fp	fn
HashSet_A	98.03%	0.00%	1.97%
HashSet_B	98.03%	0.49%	1.48%
SpecForge_HS	97.04%	0.00%	2.96%
StringToken	93.49%	2.34%	4.17%
SpecForge_ST	91.15%	3.39%	5.47%
StackAr	98.50%	1.00%	0.50%
SpecForge_SA	98.50%	0.00%	1.50%

CrowdSpec v. SpecForge

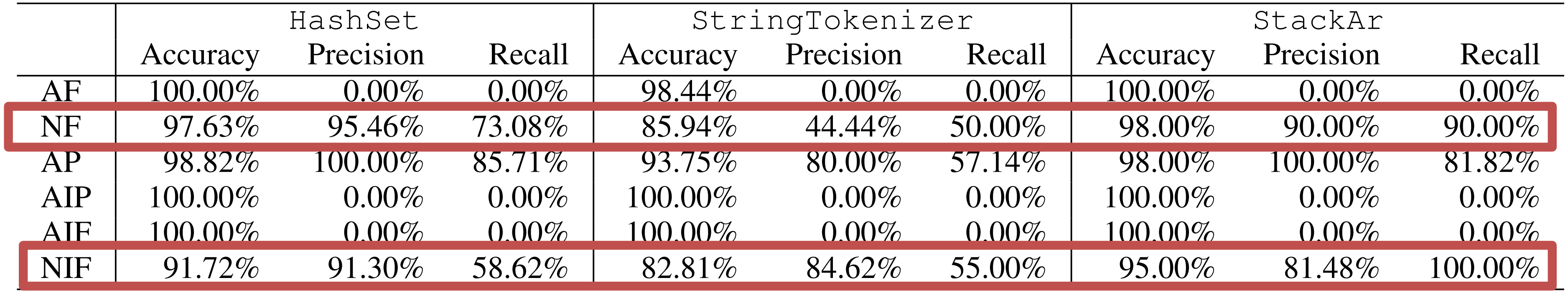
Study	Accuracy	fp	fn
HashSet_A	98.03%	0.00%	1.97%
HashSet_B	98.03%	0.49%	1.48%
SpecForge_HS	97.04%	0.00%	2.96%
StringToken	93.49%	2.34%	4.17%
SpecForge_ST	91.15%	3.39%	5.47%
StackAr	98.50%	1.00%	0.50%
SpecForge_SA	98.50%	0.00%	1.50%

- Outperform SpecForge

Results for different property types

	HashSet			StringTokenizer			StackAr		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
AF	100.00%	0.00%	0.00%	98.44%	0.00%	0.00%	100.00%	0.00%	0.00%
NF	97.63%	95.46%	73.08%	85.94%	44.44%	50.00%	98.00%	90.00%	90.00%
AP	98.82%	100.00%	85.71%	93.75%	80.00%	57.14%	98.00%	100.00%	81.82%
AIP	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%
AIF	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%
NIF	91.72%	91.30%	58.62%	82.81%	84.62%	55.00%	95.00%	81.48%	100.00%

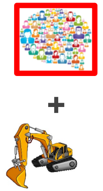
Results for different property types



	HashSet			StringTokenizer			StackAr		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
AF	100.00%	0.00%	0.00%	98.44%	0.00%	0.00%	100.00%	0.00%	0.00%
NF	97.63%	95.46%	73.08%	85.94%	44.44%	50.00%	98.00%	90.00%	90.00%
AP	98.82%	100.00%	85.71%	93.75%	80.00%	57.14%	98.00%	100.00%	81.82%
AIP	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%
AIF	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%	0.00%	0.00%
NIF	91.72%	91.30%	58.62%	82.81%	84.62%	55.00%	95.00%	81.48%	100.00%

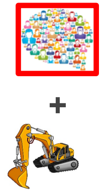
- Crowd isn’t great at *“never”* property types

Accuracy comparison



API	SpecForge	SF+ CrowdSpec	Expert1	Expert2	Expert3	Experts Voting	Experts Discussing
`HashSet`	97.04%	98.03%	99.61%	98.32%	98.22%	98.42%	100%
`StTokenizer`	91.15%	93.49%	97.14%	97.92%	98.44%	100.00%	100%
`StackAr`	98.50%	98.50%	98.17%	96.50%	98.67%	98.67%	100%

Accuracy comparison



API	SpecForge	SF+ CrowdSpec	Expert1	Expert2	Expert3	Experts Voting	Experts Discussing
`HashSet`	97.04%	98.03%	99.61%	98.32%	98.22%	98.42%	100%
`StTokenizer`	91.15%	93.49%	97.14%	97.92%	98.44%	100.00%	100%
`StackAr`	98.50%	98.50%	98.17%	96.50%	98.67%	98.67%	100%

- CrowdSpec improves SpecForge

Accuracy comparison

+

		SF+				Experts	Experts
API	SpecForge	CrowdSpec	Expert1	Expert2	Expert3	Voting	Discussing
`HashSet`	97.04%	98.03%	99.61%	98.32%	98.22%	98.42%	100%
`StTokenizer`	91.15%	93.49%	97.14%	97.92%	98.44%	100.00%	100%
`StackAr`	98.50%	98.50%	98.17%	96.50%	98.67%	98.67%	100%

- Combo gets close to voting experts

Accuracy comparison

+

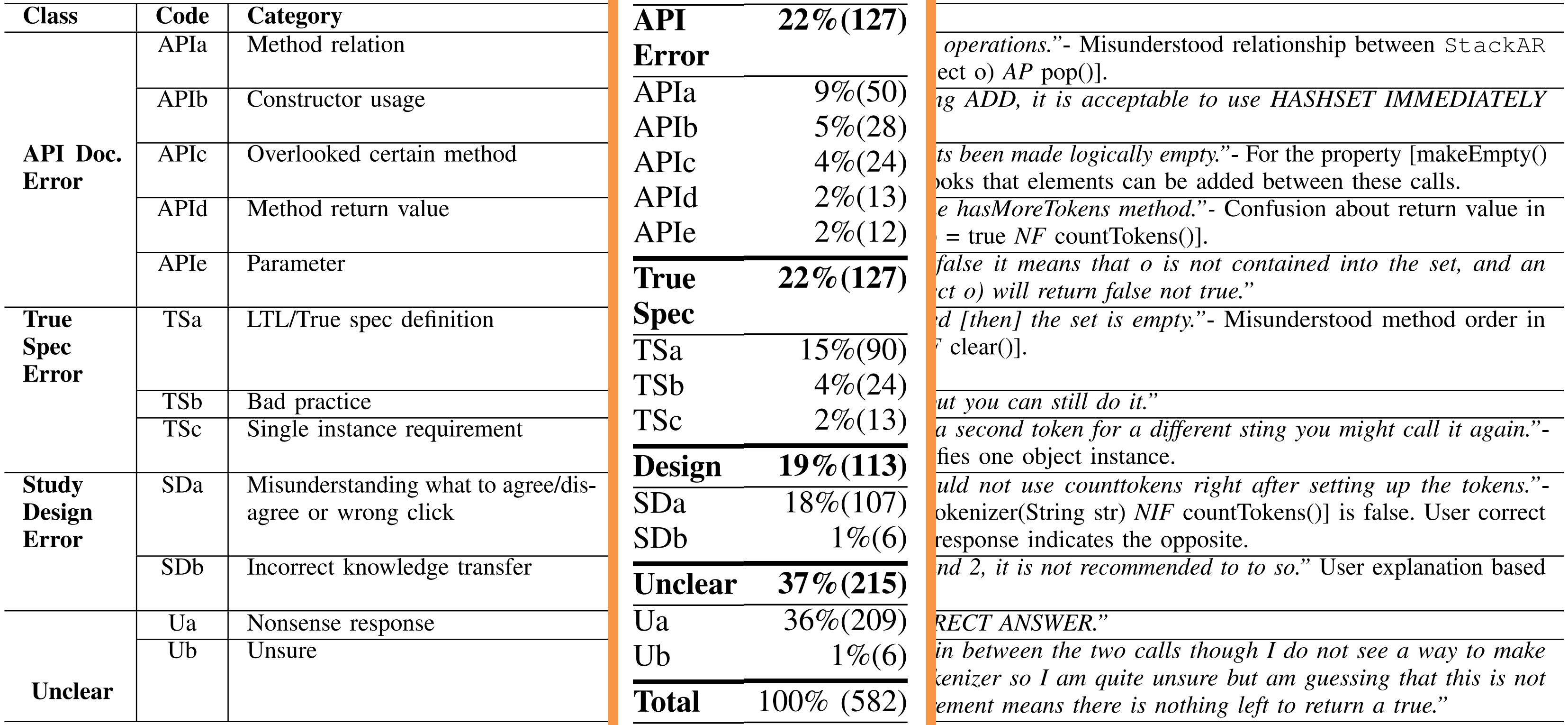
API	SpecForge	SF+ CrowdSpec	Expert1	Expert2	Expert3	Experts Voting	Experts Discussing
HashSet	97.04%	98.03%	99.61%	98.32%	98.22%	98.42%	100%
StTokenizer	91.15%	93.49%	97.14%	97.92%	98.44%	100.00%	100%
StackAr	98.50%	98.50%	98.17%	96.50%	98.67%	98.67%	100%

- But, discussing experts.. unbeatable

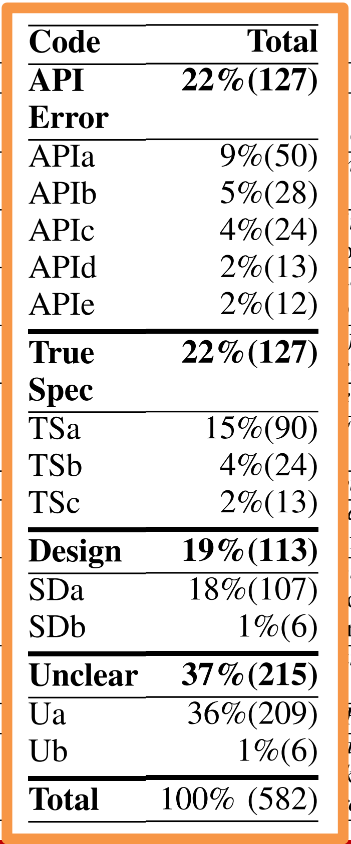
Crowd errors

Class	Code	Category	Example
API Doc. Error	APIa	Method relation	*"These are opposite, unrelated operations."*- Misunderstood relationship between `StackAR` methods in property [push(Object o) *AP* pop()].
	APIb	Constructor usage	*"In HashSet libray, when using ADD, it is acceptable to use HASHSET IMMEDIATELY afterward."*
	APIc	Overlooked certain method	*"[A] stack cannot be full after its been made logically empty."*- For the property [makeEmpty() *AF* isFull() = true], user overlooks that elements can be added between these calls.
	APId	Method return value	*"Returns the same value as the hasMoreTokens method."*- Confusion about return value in the property [hasMoreTokens() = true *NF* countTokens()].
	APIe	Parameter	*"if remove(Object o) returns false it means that o is not contained into the set, and an immediate call to remove(Object o) will return false not true."*
True Spec Error	TSa	LTL/True spec definition	*"Once all elements are cleared [then] the set is empty."*- Misunderstood method order in property [isEmpty() = true *AIF* clear()].
	TSb	Bad practice	*"Bad programming practice, but you can still do it."*
	TSc	Single instance requirement	*"Well if you wanted to create a second token for a different sting you might call it again."*- Confused about task that specifies one object instance.
Study Design Error	SDa	Misunderstanding what to agree/disagree or wrong click	*"I see no reason why you could not use counttokens right after setting up the tokens."*- Machine's answer for [StringTokenizer(String str) *NIF* countTokens()] is false. User correct reasoning, but user's property response indicates the opposite.
	SDb	Incorrect knowledge transfer	*"No, based on response on 1 and 2, it is not recommended to to so."* User explanation based on previous questions.
Unclear	Ua	Nonsense response	*"I THINK THIS IS THE CORRECT ANSWER."*
	Ub	Unsure	*"there may be changes made in between the two calls though I do not see a way to make these changes within StringTokenizer so I am quite unsure but am guessing that this is not [false] because a false measurement means there is nothing left to return a true."*

Crowd errors



Class	Code	Category	
API Doc. Error	APIa	Method relation	*operations.”*- Misunderstood relationship between `StackAR` ect o) *AP* pop()].
	APIb	Constructor usage	*ng ADD, it is acceptable to use HASHSET IMMEDIATELY*
	APIc	Overlooked certain method	*ts been made logically empty.”*- For the property [makeEmpty() ooks that elements can be added between these calls.
	APId	Method return value	*e hasMoreTokens method.”*- Confusion about return value in = true *NF* countTokens()].
	APIe	Parameter	*false it means that o is not contained into the set, and an ct o) will return false not true.”*
True Spec Error	TSa	LTL/True spec definition	*d [then] the set is empty.”*- Misunderstood method order in clear()].
	TSb	Bad practice	*ut you can still do it.”*
	TSc	Single instance requirement	*a second token for a different sting you might call it again.”*- fies one object instance.
Study Design Error	SDa	Misunderstanding what to agree/disagree or wrong click	*uld not use counttokens right after setting up the tokens.”*- okenizer(String str) *NIF* countTokens()] is false. User correct response indicates the opposite.
	SDb	Incorrect knowledge transfer	*nd 2, it is not recommended to to so.”* User explanation based
Unclear	Ua	Nonsense response	*RECT ANSWER.”*
	Ub	Unsure	*in between the two calls though I do not see a way to make kenizer so I am quite unsure but am guessing that this is not ement means there is nothing left to return a true.”*



Code	Total
API Error	**22%(127)**
APIa	9%(50)
APIb	5%(28)
APIc	4%(24)
APId	2%(13)
APIe	2%(12)
True Spec	**22%(127)**
TSa	15%(90)
TSb	4%(24)
TSc	2%(13)
Design	**19%(113)**
SDa	18%(107)
SDb	1%(6)
Unclear	**37%(215)**
Ua	36%(209)
Ub	1%(6)
Total	100% (582)

CrowdSpec take-aways

Lightweight and **scalable** approach to mine temporal specs from JavaDoc with a Crowd

- Improves existing spec-miners
- Approaches expert-level spec quality

More generally, re-consider:

- The automation dimension in your work
- SE research assumptions you can disrupt!

Our evaluation results are online: https://bestchai.bitbucket.io/crowdspecmine-eval/

Metrics

Majority rule to determine the crowd's opinion.

We measure:

- **Precision**: the percentage of properties that are actually true, of those that are reported to be true.
- **Recall**: the percentage of the true properties that are reported to be true.
- **Accuracy**: the percent of correct mined properties, true and false, in the ground truth.

		Ground Truth	
		True	False
Crowd Decision	True	True Positive (tp)	False Positive (fp)
	False	False Negative (fn)	True Negative (tn)

Distribution of true instances

Property	HashSet	StringTokenizer	StackAr
AF	0%(0)	0%(0)	0%(0)
NF	8%(13)	13%(8)	10%(10)
AP	8%(14)	11%(7)	11%(11)
AIP	0%(0)	0%(0)	0%(0)
AIF	0%(0)	0%(0)	0%(0)
NIF	17%(29)	31%(20)	22%(22)

Study characteristics

Study	HashSet_A	HashSet_B	StToken	StAr
Total cost	$473.75	$473.73	$138.68	$218.05
Duration	2 days	4 days	30 days	17 days

Study specifics

Study Features	*HashSet_A*	*HashSet_B*	*StringToken*	*StackAr*
People per task	5	5	3/4/5	3/4/5
Payment	$0.40	$0.40	$0.40	$0.40
Total cost	$473.75	$473.73	$138.68	$218.05
Valid responses	845	845	246	388
Duration	2 days	4 days	30 days	17 days

Quality Control	*HashSet_A*	*HashSet_B*	*StringToken*	*StackAr*
Qualification test	yes	yes	yes	yes
# questions	7	7	7	7
Conflict detection	yes	yes	yes	yes
Gold standard	yes	yes	yes	yes
Random click	yes	yes	yes	yes

Participants	*HashSet_A*	*HashSet_B*	*StringToken*	*StackAr*
Total participants	39	38	66	55
Male/female/unk	30/9/0	28/8/2	51/15/0	32/23/0
Avg. age	30	31	33	34
% CS degree	74%	74%	68%	60%
Java familiarity	3.87	3.95	3.64	3.51