

Engineering Iron Man's JARVIS

Dr. Emily Hill

Drew University

OOPSLE 2015

New York scientists unveil 'invisibility cloak' to rival Harry Potter's

BY CAURIE PUTNAM

ROCHESTER N.Y. | Fri Sep 26, 2014 9:43pm EDT

 Tweet

224

 Share

72

 Share this

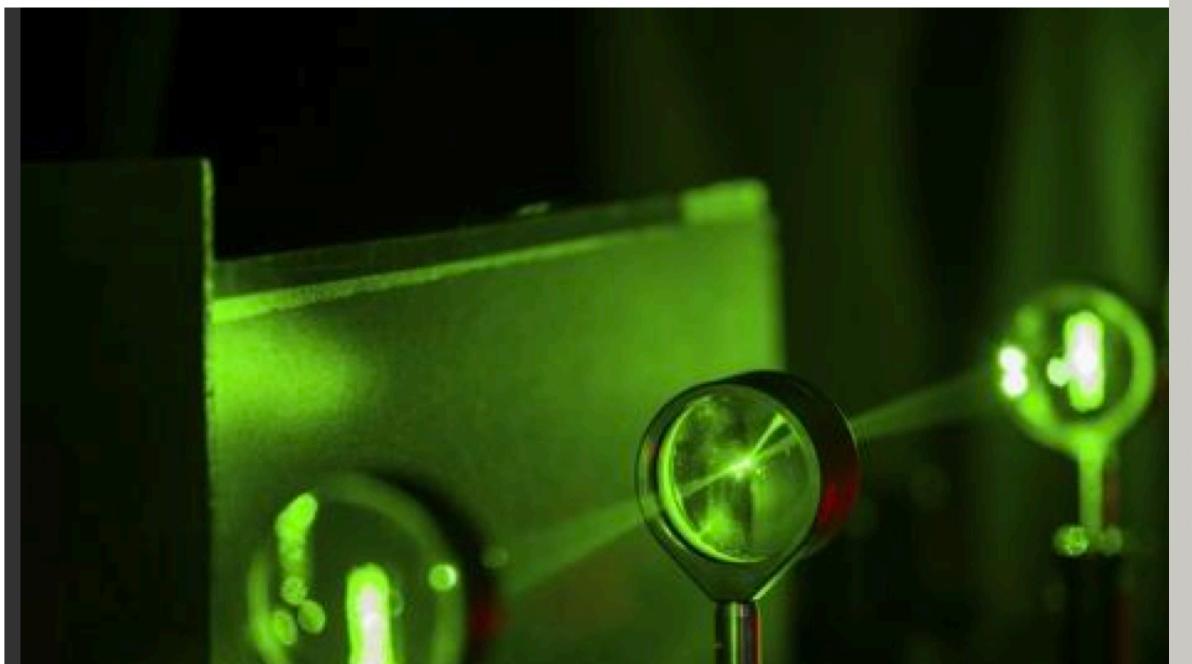
 8+1
23



Email



Print



Book and Fictional Prediction

1735 *Gulliver's Travels*

Jonathan Swift

- Mars has two moons

1865 *From the Earth to the Moon*

Jules Verne

- Lunar modules launched from Florida, return as splashdown capsules

Solar sails

1870 *Twenty Thousand Leagues Under the Sea*

Jules Verne

- Electric submarines

1888-1889 *Looking Backwards*

Edward Bellamy

- Credit cards

1903 *In the Year 2889*

Jules Verne

- Skywriting

Video Chatting "Phonotelephone"

1899 *When the Sleeper Wakes*

H.G. Wells

- Automatic motion-sensing doors

1910-1911 *The Land Ironclads*

H.G. Wells

- Tanks

1910-1911 *The Achievements of Luther Trant*

Edwin Balmer & William MacHarg

- Lie detector test

1914 *Ralph 124C 41+*

Hugo Gernsback

- Radar

Solar energy

1923-1924 *The World Set Free*

H.G. Wells

- Atomic bombs

1924 *Men Like Gods*

H.G. Wells

- Voiceicemail

1932 *Daedalus: or, Science and the Future*

J.B.S. Haldane

- In vitro fertilization

1984 *Brave New World*

Aldous Huxley

- Mood-enhancing drugs

Genetic engineering

1984 *Fahrenheit 451*

George Orwell

- Widespread invasive government spying

Actual Discovery

1877 Mars' two moons discovered

YEARS TO DISCOVERY

142

1969 1st used on Apollo 11 mission

YEARS TO DISCOVERY

184

2010 1st used successfully (IKAROS)

YEARS TO DISCOVERY

145

1960s Electric submarines invented

YEARS TO DISCOVERY

94

1950 Credit cards invented

YEARS TO DISCOVERY

62

1915 1st at San Francisco airshows

YEARS TO DISCOVERY

26

1964 - AT&T debuts Picturephone at 1964 World's Fair

YEARS TO DISCOVERY

73

1960 Automatic motion-sensing doors invented

YEARS TO DISCOVERY

68

1916 1st used

YEARS TO DISCOVERY

13

1924 1st polygraph used

YEARS TO DISCOVERY

14

1935 Radar invented

YEARS TO DISCOVERY

24

1978 1st solar-powered calculators

YEARS TO DISCOVERY

67

1945 Atomic bombs 1st used

YEARS TO DISCOVERY

31

1980 Voicemail popularized

YEARS TO DISCOVERY

57

1977 1st successful in vitro conception

YEARS TO DISCOVERY

52

1950 Antidepressants first popularized

YEARS TO DISCOVERY

16

1972 1st DNA manipulation

YEARS TO DISCOVERY

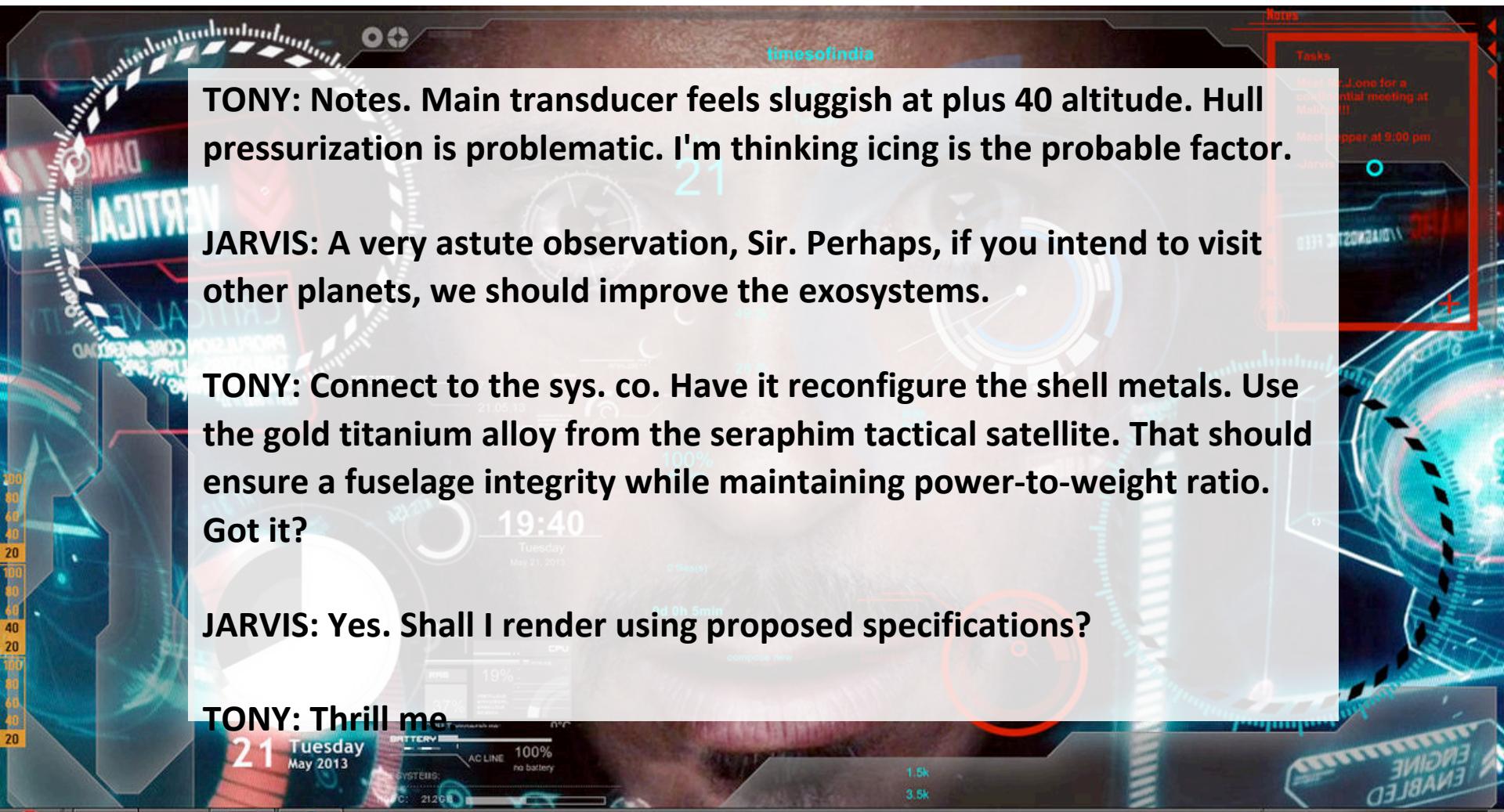
40

2013 NSA domestic spying scandal

YEARS TO DISCOVERY

65

JARVIS: Just A Rather Very Intelligent System



Programming Jarvis

- Connect to the sys. co. Have it reconfigure the shell metals. Use the gold titanium alloy from the seraphim tactical satellite. That should ensure a fuselage integrity while maintaining power-to-weight ratio. Got it?
- Commands:
 - Connect
 - Reconfigure with parameters (use) & constraints (ensure...while)
 - Confirmation

Other applications?

- Anecdotally, hardest challenge for novice programmers is translating high-level NL description of problem into 7 basic programming concepts (variables, lists, loops, functions, conditions, etc)
- Could we meet the coming shortage of programmers by making programming languages operate at a higher level?
- Or at the very least, make better programming tutors?

How close are we?

Code
Search

Naturalistic Programming
Languages [Knoll &
Mezini OOPSLA '06]

Informal Software
Representations
[Arnold & Lieberman
OOPSLA '10]

Keyword
Programming
[Little ASE '07]

Method
invocation
search

Representing NL
Phrasal Concepts
[Hill 2010]

Code
Completion

People-Specific
Languages
[Poss OOPSLE '14]

How close are we?

- Keyword Programming (Little ASE 2007)
- Code completion
- People-specific languages (Poss OOPSLE 2014)
- NLP (Hill 2010)

Naturalistic PLs

(Knoll & Mezini OOPSLA '06)

Pegasus reads ideas described in a natural language, the *input program*, and creates from it a fully executable program file, the *output program*, realizing the described ideas. For instance, Pegasus could read this natural language description:

```
Write three times: "I can understand you!" .
```

and generate the following Java program, realizing the idea of the natural language description.

```
class Main
{
    public static void main()
    {
        for (int i = 0; i < 3; i++)
            System.out.println("I can understand you!");
    }
}
```

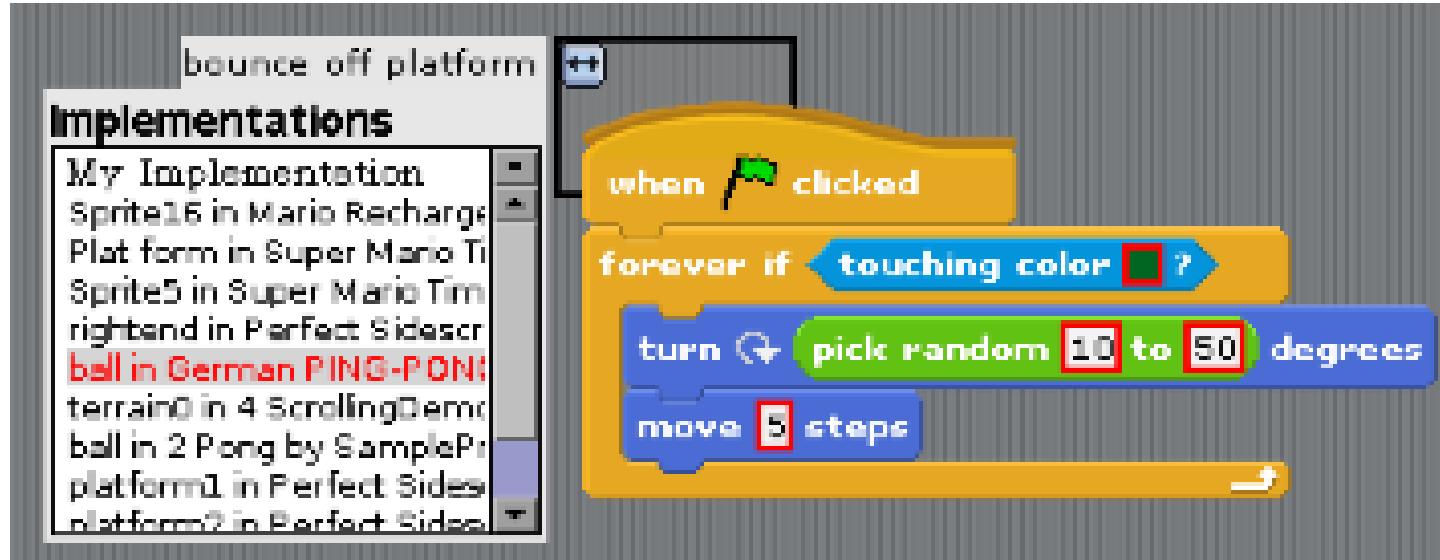
Challenges:

- **unnaturally precise wording?**
- **Dealing with ambiguity & domain knowledge**

People-Specific Languages

- Domain-specific languages taken to the extreme
- Open problems:
 - What personality/programming traits should be part of a PSL?
 - Male/female?
 - Personality features: intro vs extrovert?
 - Tool support to convert between PSLs

Informal Software Representations



(b) Given a purpose statement, the Zone sidebar (left) shows code that might fulfill it. Selecting an implementation from the list on the left shows its code on the right. As a simulation of future functionality, red boxes surround values that vary among otherwise similar code, highlighting what might need to be changed.

Phrasal Concepts in Source Code

- Phrasal concepts generalize to arbitrary phrases using 4 types of **semantic roles**:
 - action (**verb**)
 - theme (**direct object**)
 - secondary argument (**preposition + indirect object**)
 - auxiliary arguments (any remaining signature information)

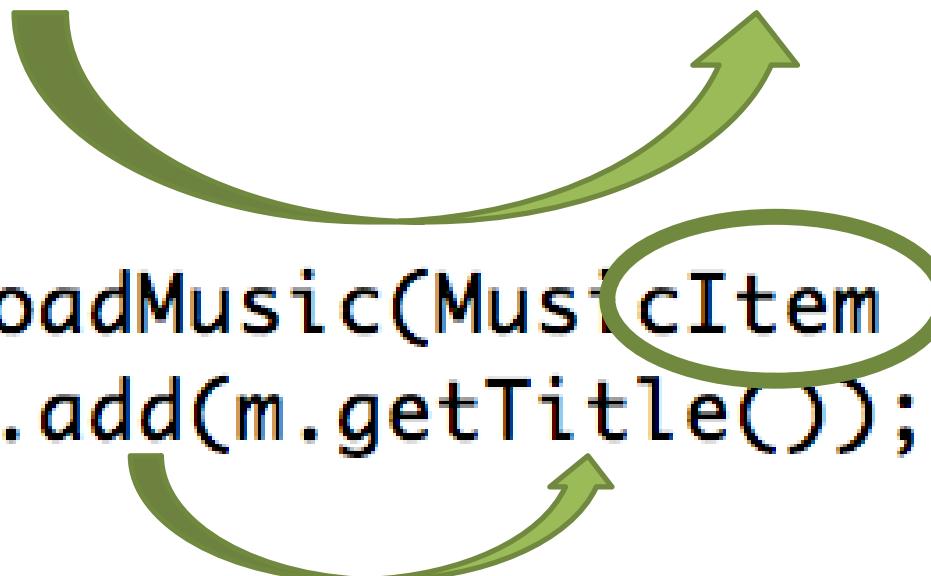
direct object (DO)
e.g., “add **item** to **list**”
indirect object (IO)

Method	action	theme (secondary arg)
SetGroupsTest.tearDown()	tear down	set groups test
Restriction.convertToMinCardinality(int)	convert	restriction (to min cardinality)
addEntry(AuctionEntry ae)	add	entry auction entry

Phrasal Concepts in Source Code

```
public void addMusic(MusicItem m) {
```

```
public void loadMusic(MusicItem m) {  
    musicList.add(m.getTitle());  
}
```

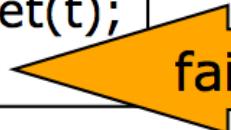


Method invocation sequences: RANDOOP

- Input:
 - classes under test
 - time limit
 - set of contracts
 - Method contracts (e.g. “o.hashCode() throws no exception”)
 - Object invariants (e.g. “o.equals(o) == true”)
- Output: contract-violating test cases. Example:

*no contracts
violated
up to last
method call*

```
HashMap h = new HashMap();
Collection c = h.values();
Object[] a = c.toArray();
LinkedList l = new LinkedList();
l.addFirst(a);
TreeSet t = new TreeSet(l);
Set u = Collections.unmodifiableSet(t);
assertTrue(u.equals(u));
```

 fails when executed

Problem Summary: Translation

- High-level NL description of problem → executable code
- Possible solutions/directions:
 - Pull out actionable VPs from NL & search for code examples or chains of invocations (a la RANDOOP)
 - Constrain NL (a la Pegasus) or train PL with people-specific DSLs
 - Learn correspondence between HL NL -> PL
 - Documentation-code mining (naming conventions & naming bugs)
 - Apply comment generation rules in reverse

End-user naturalistic programming:

Feasible? Pipe dream?

Why so little progress
from PL community?