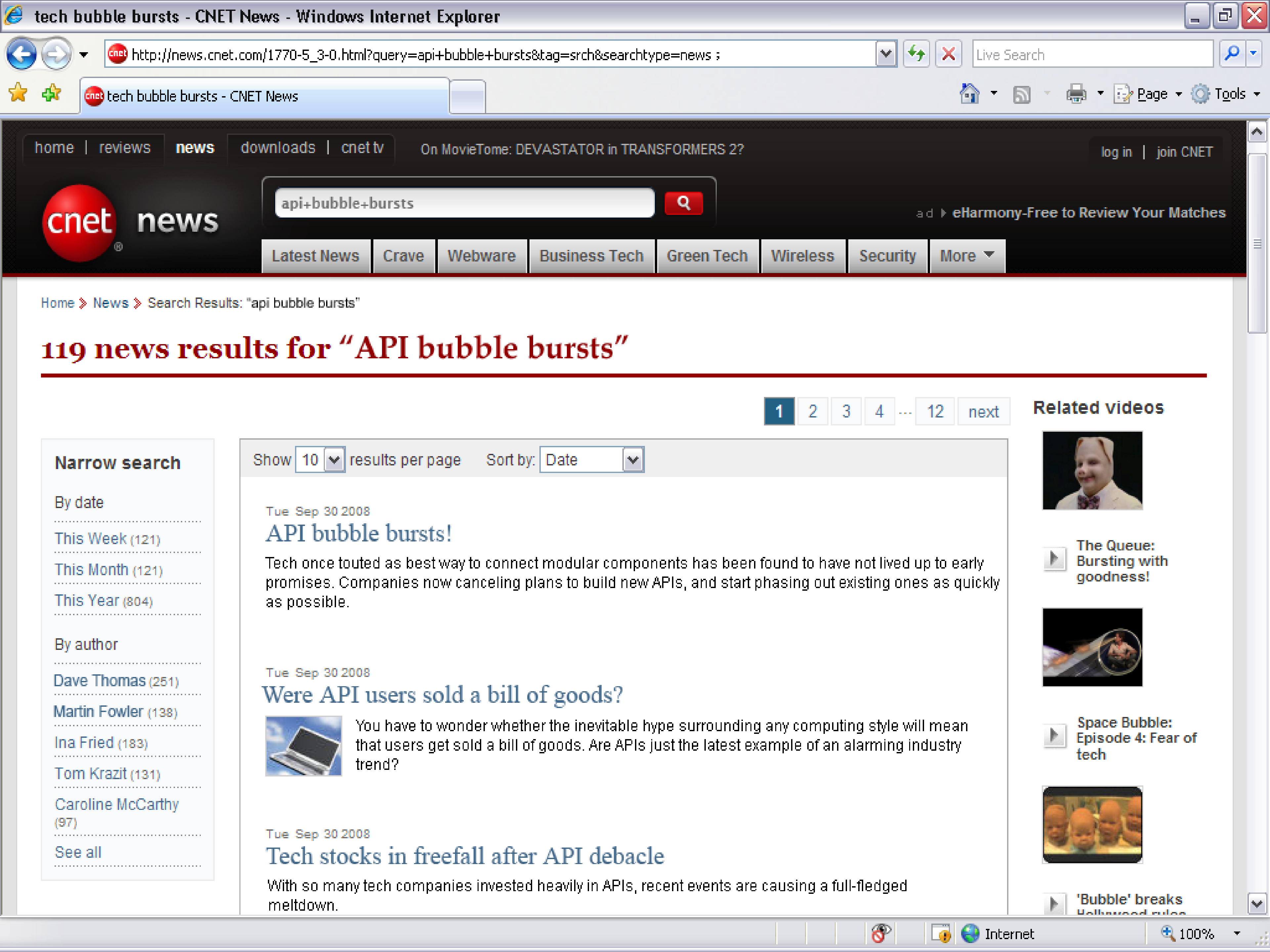
APIs - Lines in the Sand

Jim des Rivières, IBM
Jim_des_rivieres@ca.ibm.com



tech bubble bursts - CNET News - Windows Internet Explorer
http://news.cnet.com/1770-5_3-0.html?query=api+bubble+bursts&tag=srch&searchtype=news ;
Live Search
tech bubble bursts - CNET News
Page
Tools
home | reviews | news | downloads | cnet tv
On MovieTome: DEVASTATOR in TRANSFORMERS 2?
log in | join CNET
cnet news
api+bubble+bursts
ad eHarmony-Free to Review Your Matches
Latest News
Crave
Webware
Business Tech
Green Tech
Wireless
Security
More
Internet
100%

Home > News > Search Results: "api bubble bursts"

119 news results for "API bubble bursts"

1 2 3 4 ... 12 next

Narrow search

By date

- This Week (121)
- This Month (121)
- This Year (804)

By author

- Dave Thomas (251)
- Martin Fowler (138)
- Ina Fried (183)
- Tom Krazit (131)
- Caroline McCarthy (97)
- See all

Show 10 results per page Sort by: Date

Tue Sep 30 2008

API bubble bursts!

Tech once touted as best way to connect modular components has been found to have not lived up to early promises. Companies now canceling plans to build new APIs, and start phasing out existing ones as quickly as possible.

Tue Sep 30 2008

Were API users sold a bill of goods?

You have to wonder whether the inevitable hype surrounding any computing style will mean that users get sold a bill of goods. Are APIs just the latest example of an alarming industry trend?

Tue Sep 30 2008

Tech stocks in freefall after API debacle

With so many tech companies invested heavily in APIs, recent events are causing a full-fledged meltdown.

Related videos

- The Queue: Bursting with goodness!
- Space Bubble: Episode 4: Fear of tech
- 'Bubble' breaks

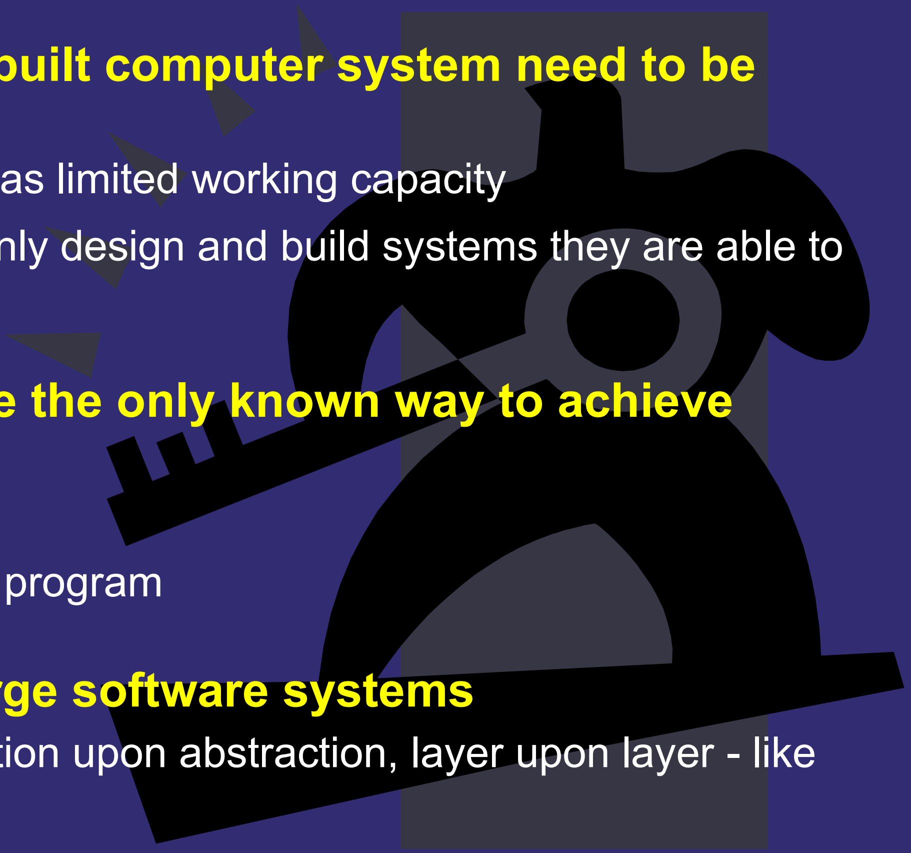
Modules, abstractions, APIs, …

These notions are

- **Ubiquitous**
- **Economically important**
- **Thriving**
- **Not new**

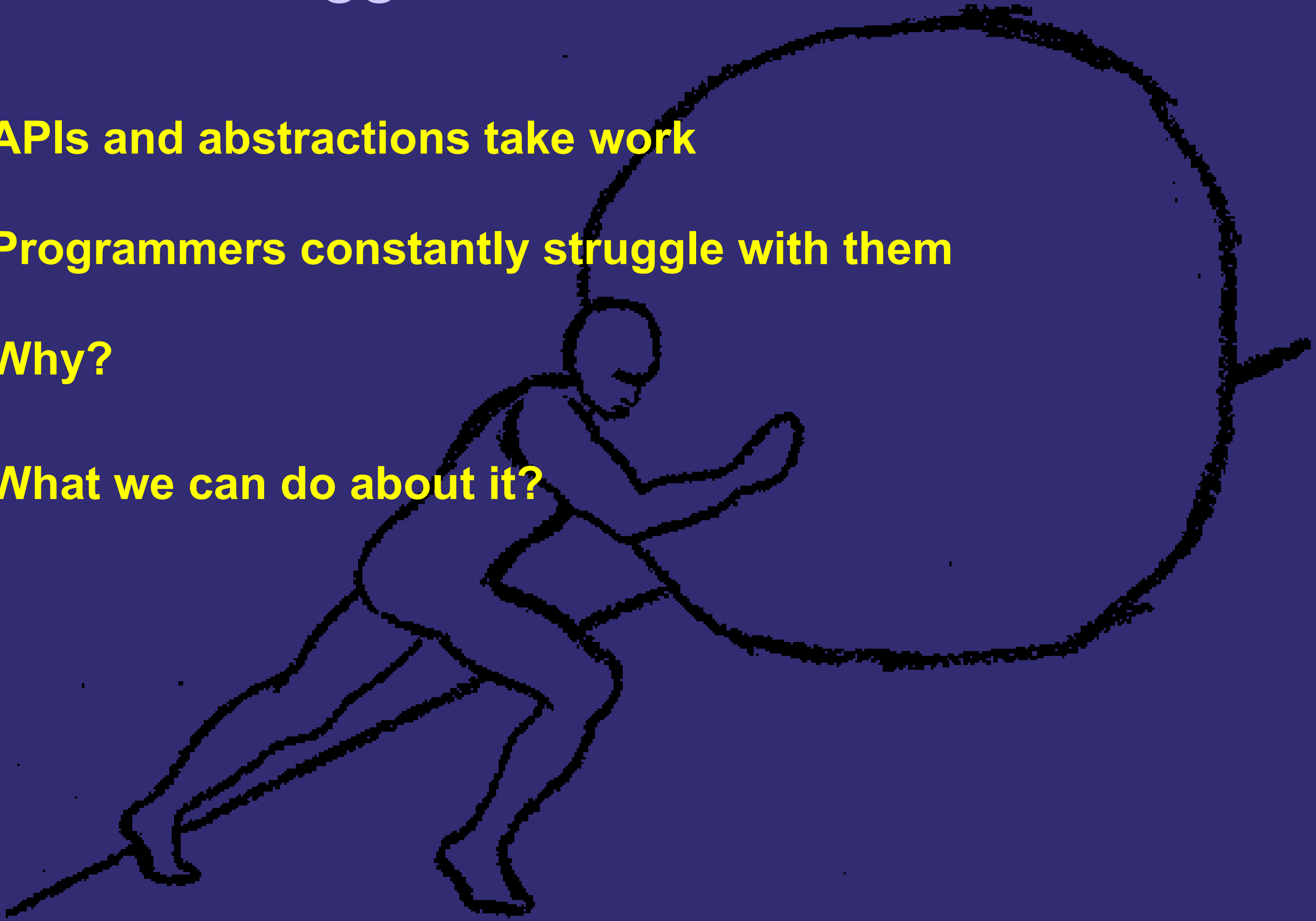
Why so important?

- **Inner working of human-built computer system need to be <u>scrutable</u>**
  - Individual human mind has limited working capacity
  - Teams of humans can only design and build systems they are able to understand
- **APIs and abstractions are the only known way to achieve scrutability at any scale**
  - Divide and conquer
  - Scales linearly in size of program
- **Essential aspect of all large software systems**
  - API stacks, built abstraction upon abstraction, layer upon layer - like coral reefs



Programmers struggle with APIs

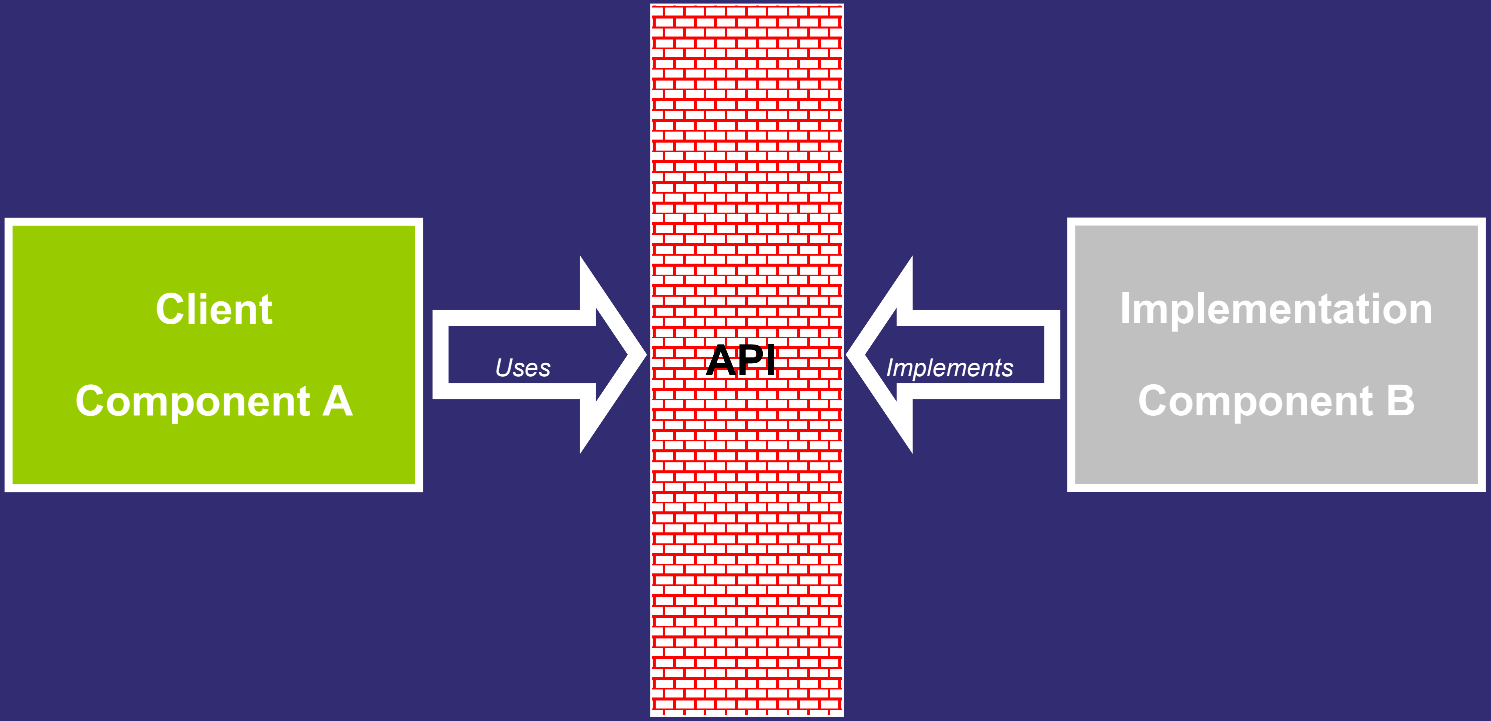
- APIs and abstractions take work
- Programmers constantly struggle with them
- Why?
- What we can do about it?



Struggles with APIs

- **Theme for talk**
- **Sampler of our struggles**
  - 1. Our tools
  - 2. Drawing API lines
  - 3. Our methodologies
  - 4. Our natures

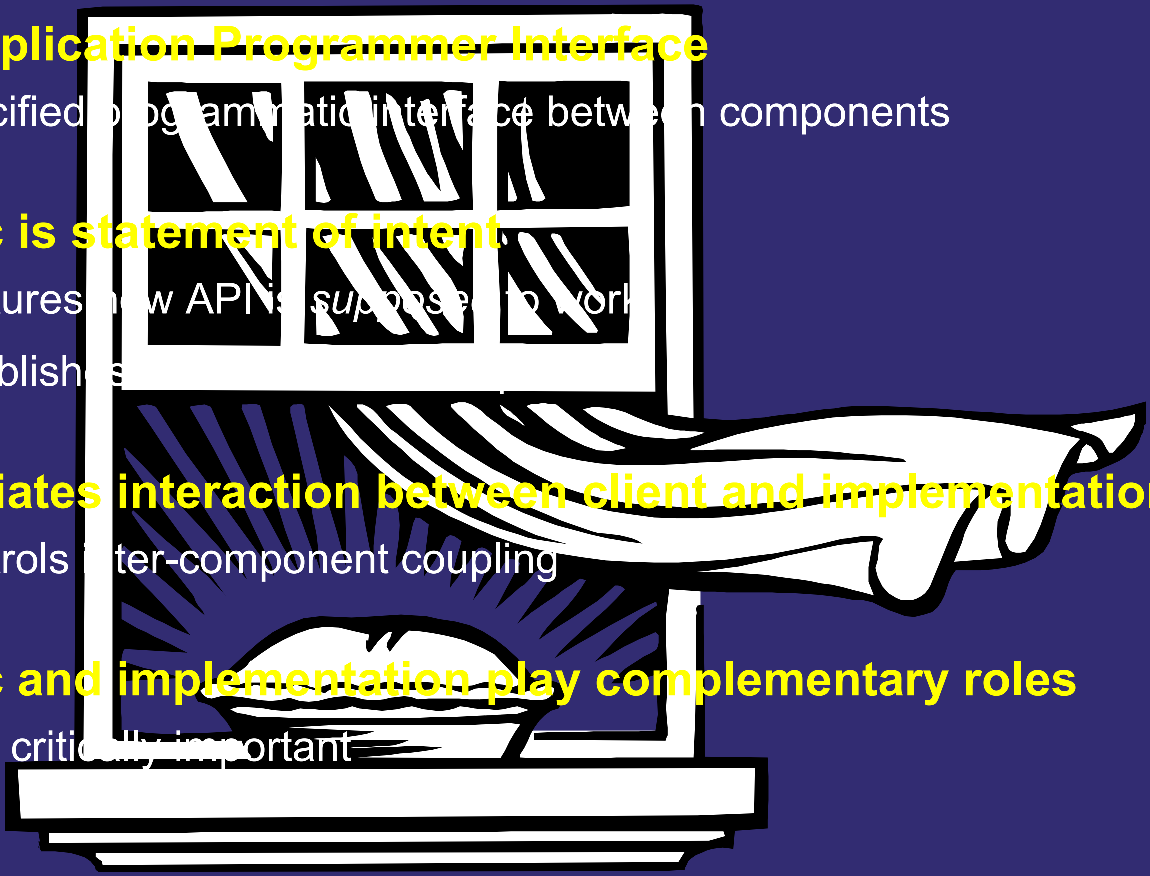
Vocabulary



- **API = <u>specified</u> programmatic interface between components**

Vocabulary

- **API = Application Programmer Interface**
  - Specified programmatic interface between components
- **API spec is statement of intent**
  - Captures how API is *supposed* to work
  - Establishes
- **API mediates interaction between client and implementation**
  - Controls inter-component coupling
- **API spec and implementation play complementary roles**
  - Both critically important



1. Struggles with APIs - Our tools

- **Relentlessly refactor code to improve scrutability**
- **Refactoring tools help us**
  - Tool has awareness of language semantics
  - Can apply meaning-preserving transformations

Bull in a china shop

- **Refactoring tools are dangerous where APIs are involved**
  - Will propose refactoring across API boundaries
  - Will propose changes to API signatures
- **Tools are blind to this important aspect of program**

What can we do about it?

- **Tools can be more helpful if they are API-aware**
- **Programmer can map out where APIs are**
- **Example: Eclipse PDE has map for plug-in's APIs**
  - Eclipse nightly build compares APIs to baseline
  - Report API changes, and classify as potentially breaking
- **Refactoring tool can use API map when proposing refactorings**

2. Struggles with APIs - Drawing API lines

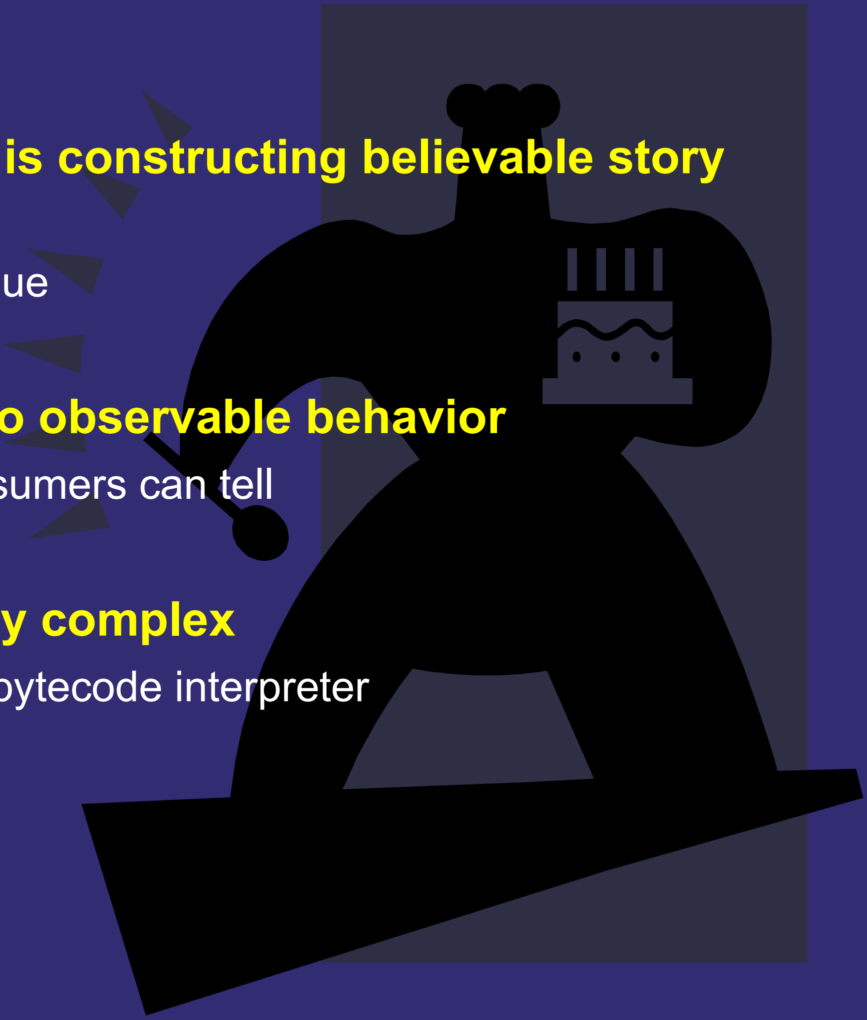
- **Common question: How to add API to existing program?**
  - Program not written with API in mind
- **They can identify a subset of classes and methods they feel comfortable letting consumers call**
  - Mark these public; mark rest private
  - Add Javadoc
- **They can write unit tests**
- **Is there anything else?**

It's not a lie. It's a gift for fiction.

- **API is like spy’s cover story in a Le Carré novel**
- **Make cover story compelling so you won’t have to reveal truth**
  - Self-consistent, no gaps, no gaffs

A Convenient Fiction

- **Designing API and writing spec is constructing believable story for consumers**
  - Consumers take story at face value
- **Implementation is constrained to observable behavior**
  - Story appears true as far as consumers can tell
- **Implementation can be arbitrarily complex**
  - E.g., JIT compiler implementing bytecode interpreter
- **Have cake and eat it too**



Mixing genres

- **Adding an API to existing program - hard**
- **Adding an API to existing program without breaking existing clients – nigh impossible**
- **Bad to let shape of program dictate shape of API**
  - API will be harder for consumers to use
  - API will not provide enough cover for implementation

What do we do about it?

- Set wad to the side temporarily
- Design API to give consumer what they really need
- Write test suites
- Cannibalize wad for implementation

- If an API is in the cards, usually cheaper to do it from outset rather than later

- API First methodology

3. Struggles with APIs - Our methodologies

- **Write program sketch – stubs for classes and methods**
- **Write unit tests that check program has desired behavior**
- **Write implementation of desired behavior**

```
Loop
  Run tests
  Exit if all tests green
  Debug program
End
```

- **To later change desired behavior**
  - Update tests to reflect behavior now desired
  - Update implementation
  - Rerun test-debug loop
- **Development method maintains invariant**
  - Program implements desired behavior
  - Test suite capable of verifying program implements desired behavior

Beating around the bush

- **Good – unit tests capitalize on programmer's skill at writing programs**
- **Not so good – intended behavior is captured implicitly**
- **How does a consumer learn of program behavior?**
  - Read program code?
  - Read test suites?
- **Fails to address what consumers need to use program**

What can we do about it?

- **Recognize central role played by API specs**
  - Spec explicitly captures intent in form consumer can use

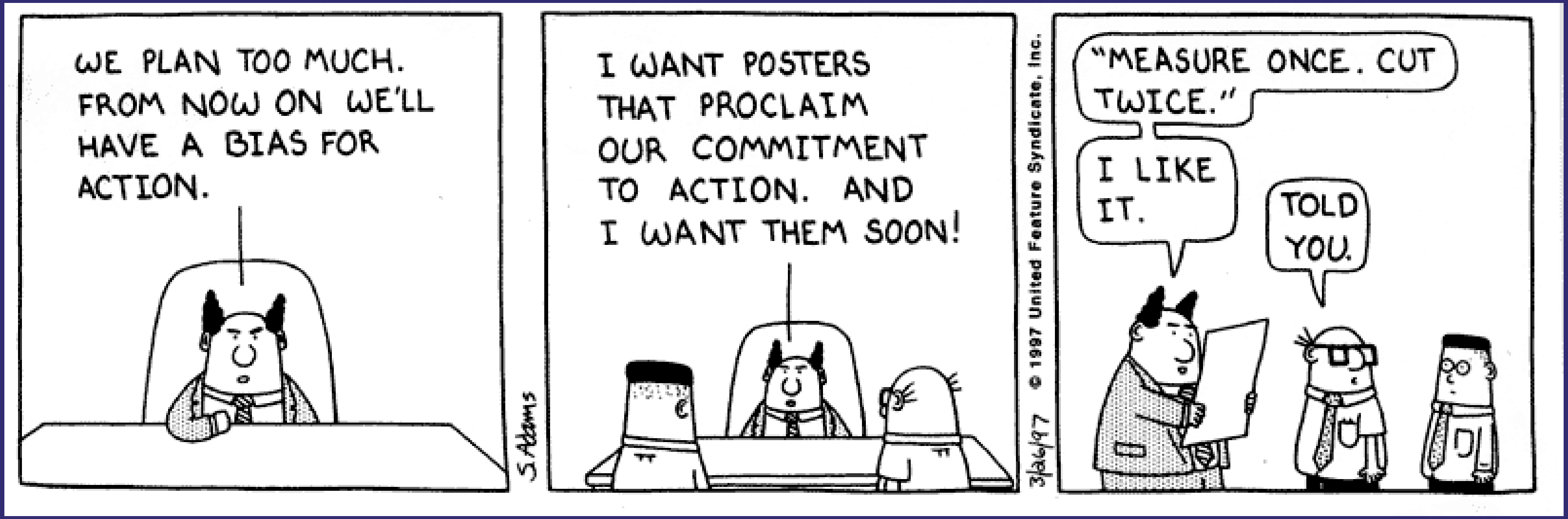
- **Design API by writing API specs (Javadoc) for classes, methods, etc.**
- **Write unit tests from specs**
- **Write implementation from specs**
- **Perform test-debug loop until tests run green**

- **To later change API**
  - Update API specs to reflect behavior now desired
  - Update unit tests and implementation based on revised specs
  - Rerun test-debug loop

- **Development method maintains invariant**
  - Full API specs providing story for consumers
  - Program implementing API to spec
  - Test suite verifying program implements API to spec

- **Test First methodology -> API First methodology**

4. Struggles with APIs - Our natures


WE PLAN TOO MUCH. FROM NOW ON WE'LL HAVE A BIAS FOR ACTION.
I WANT POSTERS THAT PROCLAIM OUR COMMITMENT TO ACTION. AND I WANT THEM SOON!
"MEASURE ONCE. CUT TWICE."
I LIKE IT.
TOLD YOU.
S. Adams
3/26/97
© 1997 United Feature Syndicate, Inc.

Programmers have natural bias for action

- **We are professional programmers**
- **We get paid to BUILD systems that DO something**

Guerilla programming

- **Task: write program to extract data for a one-time report**
  - Data is in these databases
  - Accessible through unfamiliar API
- **Deadline: tomorrow**
- **How would you go about it?**

Would you?

A. Carefully read API doc so you could use API correctly

C. Handle rare conditions that do not arise during execution

E. Write test suites

G. Design new APIs and abstractions

I. Write documentation

K. None of the above

Or would you?

- **Cobble together snippets from example programs that use API**
- **Write and debug and experiment on the fly**
- **Wing it**

Welcome

to the

Dark Side

Programmer’s psyche

- Recognize in yourself this powerful attraction
- Purely mechanistic side of programming
- Nothing except “here” and “now”
- No rules other than what works
- Scrutability is minor consideration

Commercial/industrial programming

- **What we do day-to-day as programmers is not unrelated**
  - Circumstances differ from guerilla programming
- **Program execution context**
  - “there and then” vs. “here and now”
- **Program development context**
  - Write, test, and debug out of context vs. in context
- ***Program size**
  - Large vs. small
- ***Program lifetime**
  - Long-lived vs. one-shot
- ***Development team**
  - Teams of programmers vs. solo programmer
- *** Last 3 push on scrutability**

Programmer recidivism

- **Common cause of low quality code: Programmer falls back on guerilla programming practices in context where inappropriate**
  - Insufficient familiarity with APIs they are using
  - Omit handling for rare program conditions
  - Insufficient testing
  - Unaware of API boundaries already in place
  - Insufficient concern for program scrutability

Other Symptoms

- Procrastination on defining an API
- Disengagement from writing specifications
- No process for maintaining and evolving API

What can we do about it?

- **Acknowledge that some programmers are drawn more strongly to mechanistic side of programming than others**

- **Provide training, practice, support**

- **Choose/assign programming tasks accordingly**
- **Tasks relying heavily on non-mechanistic aspects**
  - Designing and evolving APIs
- **Tasks relying more on mechanistic aspects**
  - Implementing APIs
  - Testing, debugging

Concluding remarks

- **APIs, abstractions, modules etc. are crucial to most everything we do**
  - In many cases, more lasting value in API spec that in code for implementations or clients; e.g. HTTP spec
- **APIs are “soft” properties of computer system, not “hard” (mechanistic) ones**
  - Do not appeal to programmer’s bias for action
- **Many practices, tools, methodologies are colored by same bias for action**
  - No surprise - developed by programmers for programmers
  - But they can be improved to help with APIs too
- **Challenge: Analyze your own individual and team work habits. Are you there places where a bias for action is downplaying APIs?**

Questions
?

Thank you

Some API-related Resources

- *API First*
  http://www.eclipsecon.org/2005/presentations/EclipseCon2005_12.2APIFir
- *Eclipse APIs: Lines in the Sand*
  http://www.eclipsecon.org/2004/EclipseCon_2004_TechnicalTrackPresenta
- *How to Use the Eclipse API*
  http://www.eclipse.org/articles/Article-API%20use/eclipse-api-usage-rules.
- *Effective Java*, by Josh Bloch
  http://java.sun.com/docs/books/effective/
- *Evolving Java-based APIs*
  http://eclipse.org/eclipse/development/java-api-evolution.html
- Eclipse API Central
  http://wiki.eclipse.org/API_Central