Human-Centered Methods for Improving API Usability

Brad A. Myers
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
http://www.cs.cmu.edu/~bam
bam@cs.cmu.edu
APIs

• Application Programming Interfaces
• Includes: libraries, toolkits, frameworks, software development kits (SDKs), etc.
• Today: web services, “middleware”
• Also: internal APIs for large software systems
• Provides some functionality for reuse by other developers
Stakeholders & Their Goals

- [Myers, Stylos, CACM, 2016]
Why APIs?

• Some design goals for APIs:
  • Information Hiding – hide implementation
  • Provide device independence
  • Enable future changes to low level without requiring changes to application code
  • Protection of critical resources
  • Consistency for product consumer – toolkit can provide commonality
  • More robust code – toolkit implemented correctly
  • Run-time efficiency: provide services in an efficient way
  • Code reuse: Provide useful services only once
    • Programmer Productivity
  • …

My Goal: Allow API usability to be a first-class quality metric considered by API designers
APIs are Important and Valuable

- [www.programmableweb.com](http://www.programmableweb.com) – 17,508 APIs
- Apigee says 77% of companies rating APIs “important” to making their systems and data available
- Total market for API Web middleware was $5.5 billion in 2014
- Google recently bought Apigee for $625M
Why Apply Human-Centered Techniques?

• Programming is a human activity
  • Take the human into account
• “DevX” – like UX for User Experience
• APIs are the “interface” between the programmer and the functionality
• Design should be close to user’s plan
  • “Programming is the process of transforming a mental plan into one that is compatible with the computer.”
    — Jean-Michel Hoc
• Closeness of mapping — Green and Petre
• If an API cannot be used effectively by developers, it doesn’t work!
  • Even if it provides the right functionality
• Using APIs incorrectly has resulted in bugs and security problems
• Usability and quality are key influencers for the decision about which APIs to use
“Human Centered Methods” – More Than Just Lab User Studies

- Contextual Inquiry
- Contextual Analysis
- Paper prototypes
- Think-aloud protocols
- Heuristic Evaluation
- Affinity diagrams
- Personas
- Wizard of Oz
- Task analysis
- A/B testing
- Cognitive Walkthrough
- Cognitive Dimensions
- KLM and GOMS (CogTool)
- Video prototyping

- Body storming
- Expert interviews
- Questionnaires
- Surveys
- Interaction Relabeling
- Log analysis
- Storyboards
- Focus groups
- Card sorting
- Diary studies
- Improvisation
- Use cases
- Scenarios
- “Speed Dating”
- …
Human Centered Approaches Across the Lifecycle

Exploratory Studies
- Contextual Inquiries
- Interviews
- Surveys
- Lab Studies
- Corpus data mining

Design Practices
- “Natural programming”
- Graphic & Interaction Design
- Prototyping

Field Studies
- Logs & error reports

Evaluative Studies
- Expert analyses
- Usability Evaluation
- Formal A/B Lab Testing

Myers, Ko, LaToza, Yoon. IEEE Computer, 2016
HCI Techniques We Have Used for APIs

- "Contextual Inquiry" & Field Studies
  - What are the real problems & barriers that developers face?

- "Natural Programming Elicitation"
  - Let programmers express how they expect the functionality to be provided
  - How should this API be designed?

- Expert analyses
  - What are some potential problems with this API?
  - Heuristic Analysis – evaluate based on guidelines
  - Cognitive walkthrough – how hard will this specific task be to learn?

- Lab studies of programmers using an API
  - Does my API work for programmers?
  - What problems do the target developers have with my API?
  - Is this design better than that one?
“Natural Programming” Elicitation

- Technique developed by my group to elicit developer’s “natural” expressions
  - Mental models of tasks, vocabulary, etc.
- Blank paper tests
- Must prompt for the tasks in a way that doesn’t bias the answers
- Examples:
  - API Architecture
  - Words used
  - Which methods are on which classes
Context: Natural Programming Project

• Researching better tools for programming since 1978
• Natural Programming project started in 1995
• Make programming easier and more correct by making it more natural
  • Closer to the way that people think about algorithms and solving their tasks
• Methodology – human-centered approach
  • Perform studies to inform design
    • Provide new knowledge about what people do and think, & barriers
    • Guide the designs from the data
      • Design of programming languages and environments
  • Iteratively evaluate and improve the tools
• Target novice, expert and end-user programmers
End User Programming

- People whose primary job is not programming
- [Scaffidi, Shaw and Myers 2005]
  - 90 million computer users at work in US
  - 55 million will use spreadsheets or databases at work (and therefore may potentially program)
  - 13 million will describe themselves as programmers
  - 3 million professional programmers
- All of these people use APIs!
Goal: Gentle Slope Systems

Difficulty of Use

Program Complexity and Sophistication

Low Threshold

High Ceiling

Web Development
Java
Visual Basic
Flash
Server-side
C or C# Programming
JavaScript
ActionScript
CSS & HTML
Basic

© 2017 – Brad A. Myers
Our Studies of APIs

• Our work started with Jeff Stylos’s PhD, 2005-2009
  • Interned in Microsoft’s API Usability group with Steven Clarke, et. al.

• Which programming patterns are most usable?
• Measures: learnability, errors, preferences

• Studied:
  • Required parameters in constructors
  • Factory pattern
  • Object design
  • SAP’s APIs

• Tools to help with APIs
• New work: API designers’ needs
What can be addressed?

• All API design decisions

• Tools & documentation for APIs

[Stylos & Myers, VL/HCC ‘2007]
Required Constructors Study

• Compared `create-set-call` (default constructor)

```javascript
var foo = new FooClass();
foo.Bar = <get a bar>;
foo.Use();
```

• vs. required constructors (`immutable` classes):

```javascript
var tempBar = <get a bar>;
var foo = new FooClass(tempBar);
foo.Use();
```

• All participants assumed there would be a default constructor
• Required constructors interfered with learning
  • Users wanted to experiment with what kind of object to use first
• Preferred to *not* use temporary variables
• Tradeoff with the security and reliability of `immutable classes`
  • See [Coblenz, Nelson, Aldrich, Myers, Sunshine: “Glacier: Transitive Class Immutability for Java”], Wed @ 11:00
“Factory” Pattern Study

• Instead of “normal” creation: `Widget w = new Widget();`
• Objects must be created by another class:
  ```java
  AbstractFactory f = AbstractFactory.getDefault();
  Widget w = f.createWidget();
  ```
• Used frequently in Java (>61) and .Net (>13) and SAP
• Results:
  • When asked to design on “blank paper”, no one designed a factory
  • Time to develop using factories took 2.1 to 5.3 times longer compared to regular constructors (20:05 v. 9:31, 7:10 v. 1:20)
  • All subjects had difficulties getting using factories in APIs

[Ellis. Stylos & Myers, ICSE ‘2007]
Object Method Placement Study

- Where to put functions when doing object-oriented design of APIs when multiple classes work together
  - `mail_Server.send(mail_Message)` vs. `mail_Message.send(mail_Server)`
- When desired method is on the class that they start with, users were between 2.4 and 11.2 times faster (p < 0.05)
- Starting class can be predicted based on user’s tasks
- More general terms should be used most commonly
  - `Mail` vs. `Mail_server` class
  - `Java File` class

![Time to Find a Method](chart.png)

© 2017 – Brad A. Myers
Study of APIs for SAP

• Study APIs for Enterprise Service-Oriented Architectures ("Web Services")

• Naming problems:
  • Too long
  • Not understandable
  • Differences in middle are frequently missed
eSOA Documentation Results

- Multiple paths: unclear which one to use
- Some paths were dead ends
- Inconsistent look and feel caused immediate abandonment of paths
- Hard to find required information
- Business background helped
- Many other studies have reported documentation problems
Usability study of an API from SAP

- Jeff Stylos as summer intern at SAP
- SAP “Business Rules Framework Plus” API (BRFplus)
- Interviews with users
  - Identified a mismatch of abstraction level
  - API was very flexible, but users had simple use cases
- Natural programming techniques to identify expected designs
- User studies of redesigned APIs
  - Showed were successful
- Three months total work

[Stylos, Busse, Graf, Ziegler, Ehret, Karstens, VL/HCC’2008]
Evaluation based on Guidelines

- Nielsen’s Heuristics, Cognitive Dimensions
- Also Cognitive Walkthroughs
- Example: *consistency* violation:

```java
void writeStartElement(String namespaceURI,
   String localName);
void writeStartElement(String prefix,
   String localName,
   String namespaceURI);
```

*javax.xml.stream.XML.StreamWriter – [Rama, Kak, 2013]*
SAP’s NetWeaver® Gateway Developer Tools

- Plug-in to Visual Studio 2010 for developing using certain SAP APIs
- We used the HCI methods of "heuristic evaluation" and "cognitive walkthroughs" to evaluate early prototypes
- Our recommendations were quickly incorporated due to agile software development process
Automated Tools (by Others)

• Apply nine metrics to APIs
  • E.g., consistency; avoid lists of strings; factory pattern; generic exceptions; …

• API Concepts Framework takes uses of APIs into account
  [Scheller, T. and Kuhn, E. Automated measurement of API usability: The API concepts framework. *Information and Software Technology* 61 (May 2015), 145–162]
  • Interface Complexity; Implementation Complexity; Setup Complexity
Our Tools to Help with APIs

• If cannot change API, then fix the documentation and tools

• Mica

• Jadeite

• Calcite

• Dacite

• Euklas

• Graphite

• Apatite
Mica Tool to Help Find Examples

- **MICA:** Makes Interfaces Clear and Accessible
- Use Google to find relevant pages
- Match pages with Java keywords
- Also notes which pages contain example code or definitions

© 2017 – Brad A. Myers
Jadeite: Improved JavaDoc

• **JADEITE:** Java API Documentation with Extra Information Tacked-on for Emphasis
  • [http://www.cs.cmu.edu/~jadeite](http://www.cs.cmu.edu/~jadeite)
  • Mine the web for usage of Java APIs
  • Fix JavaDoc to help address problems
    • Focus attention on most popular packages and classes using font size
    • “Placeholders” for methods that users want to exist
    • Automatically extracted code examples for how to create classes
    • Related classes

See Also (auto-generated):
  Transport
  MimeMessage
  InternetAddress

Abstract void saveChanges ()
  Save any changes made to this message into the message-store when the containing folder is closed, if the message is contained in a folder.

void send ()
  Use the Transport send(message) method to send Messages

protected void setExpunged (boolean expunged)
  Sets the expunged flag for this Message.

Most common way to construct:
  SSLSocketFactory factory = ...;
  String host = ...;
  int port = ...;
  SSLSocket socket = (SSLSocket)factory.createSocket(host, port);
  Based on 38 examples
Calcite: Eclipse Plugin for Java

• **CALCITE**: Construction And Language Completion Integrated Throughout Eclipse
  • [http://www.cs.cmu.edu/~calcite](http://www.cs.cmu.edu/~calcite)

• UI = Code completion in Eclipse since familiar and usable

• Code completion in Eclipse augmented with Jadeite’s information

• How to create objects of specific classes?
  $$\text{SSLSocket } s = \_\_\_?$$
Dacite: API Designer Annotates

- **DACITE**: Design Annotations for Complementing Interfaces Targeting Effectiveness
- Visiting Professor André L. Santos from University Institute of Lisbon, Portugal
- Use Java annotations to declare properties of APIs
  - Instead of needing to search the web for them
  - More accurate & works for APIs with small user bases
- Processed by Eclipse plugin to help with API discoverability
- Unifies what Calcite did through crawling the web:
  - Supports static factories, factory methods, object builders, helper methods
  - Also adds additional patterns: decorators and composite classes
  - API designers know better what should be annotated
- Lab user study showed effective
  - Twice as many tasks finished

© 2017 – Brad A. Myers

[Santos, Myers, Journal of Systems & Software, April, 2017]
Euklas: Eclipse Plugin for JavaScript

- **EUKLAS**: Eclipse Users’ Keystrokes Lessened by Attaching from Samples
  - http://www.cs.cmu.edu/~euklas
- Postdoc Christian Dörner
- Brings Java-like analysis to JavaScript
- People often copy from examples in documentation
- Auto-correct uses copy source context for errors due to copy & paste

© 2017 – Brad A. Myers
Graphite: Eclipse Plugin for Literals

- **GRAPHITE**: **GRA**phical **P**alettes **H**elp **I**nstantiate **T**ypes in the **E**ditor.
- Pop up a custom palette for specialized constants (literals) in Eclipse
  - Regular expressions
  - Color palettes
  - Customizable

© 2017 – Brad A. Myers
Apatite Documentation Tool

• **APATITE**: Associative Perusing of APIs That Identifies Targets Easily
  - http://www.cs.cmu.edu/~apatite
  - Start with verbs (actions) and properties and find what classes implement them
  - Find associated items
    - E.g., classes that are often used together
    - Classes that implement or are used by a method

© 2013 – Brad A. Myers
New Project

- Funded by a grant from Google
- Interview and survey API designers
  - Processes used
  - Barriers to high usability
  - Information needs about API users
  - Appropriate signals of API usability
- Preliminary discussions with Google, IBM, Amazon, Bloomberg, Microsoft
  - Appear to have quite different processes
  - Different levels of sensitivity to API Usability
- Starting next week!
Open Challenges

• What other design patterns in APIs are problematic or beneficial for usability?
• How to make coordinating multiple APIs easier?
• What other design or evaluation methods are needed?
Open Challenges

• Identify **best practices** in API Design
  • How to insure that usability is a **key quality metric** that API designers always consider?
  • What **process** results in the most usable APIs?
    • What kinds of testing should be done on APIs for usability?
    • How should the API design team be organized?
  • What **guidelines** should be followed?
    • Does having guidelines even work?
    • Level of specificity of guidelines?
      • E.g., Smith and Mosier’s 1986, 486 pages of guidelines, vs. Nielsen’s 10
A Few Resources

• Brad A. Myers and Jeffrey Stylos, "Improving API Usability", *Communications of the ACM*, vol 59, No. 6, June, 2016, pp. 62-69, official ACM DL entry; html or local pdf.

• www.apiusability.org

• http://www.cs.cmu.edu/~NatProg/apiusability.html

• https://www.programmableweb.com/
Acronyms are fun!
And there are lots of Gemstones!!

For more, see: [www.cs.cmu.edu/~bam/acronyms.html](http://www.cs.cmu.edu/~bam/acronyms.html)
Thanks to:

• Funding:
  • NSF under IIS-1116724, IIS-0329090, CCF-0811610, IIS-0757511 (Creative-IT), ITR CCR-0324770 as part of the EUSES Consortium
  • SAP
  • Adobe
  • IBM
  • Microsoft Research
  • Yahoo! InMind
  • Google

• >38 students & visitors:
  ● Htet Htet Aung
  ● Jack Beaton
  ● Ruben Carbonell
  ● John R. Chang
  ● Kerry S. Chang
  ● Polo Chau
  ● Luis J. Kota
  ● Michael Coblenz
  ● Christian Dörner
  ● Dan Eisenberg
  ● Brian Ellis
  ● Andrew Faulring
  ● Aristiwidya B. (Ika) Hardjanto
  ● Erik Harpstead
  ● Amber Horvath
  ● Sae Young (Sophie) Jeong
  ● Mary Beth Kery
  ● Andrew Ko
  ● Thomas LaToza
  ● Joonhwan Lee
  ● Toby Li
  ● Leah Miller
  ● Steven Moore
  ● Gregory Mooty
  ● Yoko Nakano
  ● Stephen Oney
  ● John Pane
  ● Sunyoung Park
  ● Michael Puskas
  ● Chotirat (Ann) Ratanamahatana
  ● André L. Santos
  ● Christopher Scaffidi
  ● Jeff Stylos
  ● David A. Weitzman
  ● Yingyu (Clare) Xie
  ● Zizhuang (Zizzy) Yang
  ● YoungSeok Yoon
Human-Centered Methods for Improving API Usability

Brad A. Myers
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
http://www.cs.cmu.edu/~bam
bam@cs.cmu.edu