Usability Implications of Requiring Parameters in Objects’ Constructors

Jeffrey Stylos
Carnegie Mellon

Steven Clarke
Microsoft

ICSE 2007
May 23, 2007
What We Studied

```
mail = new Mail();
mail.setFrom(fromAddress);
mail.Send();
```

Create-Set-Call

vs APIs with

```
mail = new Mail(fromAddress);
mail.Send();
```

Required Constructor
What We Found

• API designers expected required constructor APIs to be “better,” but we found the Create-Set-Call APIs to be preferable instead.
Why We Care

• Using APIs is becoming a larger and more problematic part of programming

• Creating lots of new APIs

• Want to know how to create APIs that will be easier to use and make programmers more productive
The Problems

- APIs are hard to use
- API design is hard to get right
Our Approach

• By better understanding the impact of API design decisions we can create more usable APIs

• This study looks at constructor parameters
Prior Work

- Steven has had success studying the usability of specific APIs
- Run studies early to try to make the APIs better
- Real programmers, typical tasks
- Improve future APIs
Studying API Patterns

• Instead of studying specific APIs (e.g. a Mail API), can we study common patterns that are used across APIs?
Why Study Constructors?

- Constructor parameters are a common issue that comes up in API design
- From Steven’s other studies, he suspected that something was going on
Expected Benefits

```java
mail = new Mail();
mail.setFrom(fromAddress);
mail.Send();
```

Create-Set-Call:

Flexibility

```java
mail = new Mail(fromAddress);
mail.Send();
```

Required Constructor:

Prevent misuse; guide user
Study Method

- 30 professional programmers in a usability lab
- Representative of different “personas”
- Comparative study using several different APIs in different languages
- Observe programmers’ expectations, preferences and success using APIs
Microsoft Personas

- “Opportunistic” programmers
  - Just want it to work
- “Pragmatic” programmers
  - Want to tweak it a bit
- “Systematic” programmers
  - Want to understand everything
Programming Tasks

- Six small programming tasks
- Two versions of each task: create-set-call and required constructor
- Creating, reading, debugging code
- In Visual Studio, Notepad, on paper
Study Tasks

• Create code against imaginary APIs
• Create code against real mail and file APIs
• Create code against unfamiliar (new) APIs
• Debug code that uses real APIs
• Understand code that uses real APIs
Constructor Expectations

• “In Notepad, write code that you would expect would read in the contents of a text file and send it in the body of an email message.”

• Nearly everyone wrote code that used Create-Set-Call
Code Creation Tasks

- Opportunistic and Pragmatic programmers tried to write code using Create-Set-Call
- Opportunistic programmers had difficulty using APIs with required constructors
Code Creation Tasks

- Opportunistic programmers assumptions were so strong that they misunderstood what the problem was
- Satisfying required constructors interrupted their workflow
Programmer Workflow

- What Class Do I Use?
  - Intellisense
  - Object browser
  - Search

- Is This the Right Class?
- What Properties or Methods Do I Need?
  - Intellisense

- Instantiate the object
- Call the method / Set the property

- Do I Have to Do Anything Else?
- What's the Next Step?
  - Intellisense
What Class Do I Use?

• IntelliSense
• Object browser
• Search

Satisfy Required Constructor

Instantiate the object

Is This the Right Class?
What Properties or Methods Do I Need?

Call the method / Set the property

Do I Have to Do Anything Else?
What's the Next Step?

• IntelliSense
• IntelliSense
Required Constructors Didn’t Prevent Errors

- Programmers try to fool constructors by passing null or empty objects
- Systematic Programmers: “You still have to check for invalidity.”
- Didn’t save them any work
- Preferred being able to initialize one property at a time
Code Understanding and Debugging

- No significant difference in code debugging or code understanding
- We expected Create-Set-Call to be more readable, but IDE features helped constructor code
Conclusions

• Required Constructors didn’t prevent object mis-use as had been expected

• Required Constructors didn’t “guide” users as had been expected
Conclusions

- All programmers were more successful with create-set-call APIs
- Faster, Fewer errors, Matched expectations
- We can help API designers create more usable APIs
Ongoing Work

- Studying other API design choices
- Factory Pattern in APIs (ICSE 2007)
- (Tomorrow, 2pm)
Thanks!

jsstylos@cs.cmu.edu