Usability Implications of Requiring Parameters in Objects' Constructors

written by Jeffrey Stylos and Steven Clarke
published at ICSE 2007

talk by Dino Wernli
Intent of Paper

- Inspect influence of constructors on API usability
  - Default constructor vs. required constructor
- Perform study to analyze programmers at work
- Draw conclusions for future API design
Default Constructor Approach

- Also called "create-set-call" - approach
  - Allows objects to be created and then initialized
  - Uniform construction approach among all objects
- Use exceptions at call time if necessary

Default constructor example

```java
FooClass foo = new FooClass();
foo.setBar(barValue);
try {
    foo.use();
} catch (NoBarException e) {
    // react appropriately
}
```
Required Constructor Approach

- Only provide specific constructors
  - Method of ensuring proper initialization
  - Can guide the programmer to correct use
- Less flexible, but helps enforce invariants

```java
FooClass foo = new FooClass(barValue);
foo.use();
```

Required constructor example
Required vs. Default Constructor

- Compile time vs. runtime handling

```java
public class Socket {
    public Socket(String host) {
        this.setHost(host);
    }
    private String host;
    ...
    public void connect() {
        ...
    }
}
```

```
public class Socket {
    public Socket() {
        ...
    }
    private String host;
    ...
    public void connect() throws NoHostException {
        ...
    }
}
```

Socket with required constructor

Socket with create-set-call
Conjecture that required constructors
  - ... create more usable and self-documenting APIs
  - ... guide programmers toward correct use
  - ... are therefore the correct approach for APIs
Study Methodology

- Recruit 30 professional programmers
- Give them programming tasks
  - Using APIs with default / required constructors
  - Debugging code written in such APIs
- Gather audio and video material
- Interview participants after the tasks
Study Setup

- Room with one-way mirror
- Cameras, audio recorder
- Instructions to think out loud
Programming Personas

- Observation: different styles of programming
- Persona reflects work style **only**, not programming proficiency
- 3 different personas were analyzed in this study
3 Programming Personas

- Systematic programmer
  - Top down, understand system as a whole

- Pragmatic programmer
  - Bottom-up, but switch to top-down on fail

- Opportunistic programmer
  - Get code working as quickly as possible
  - Don't understand more than necessary
## Selection Procedure

<table>
<thead>
<tr>
<th></th>
<th>Systematic</th>
<th>Pragmatic</th>
<th>Opportunistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Experience</strong></td>
<td>&gt; 5 years</td>
<td>&gt; 2 years</td>
<td>&gt; 2 years</td>
</tr>
<tr>
<td><strong>Main language</strong></td>
<td>C / C++</td>
<td>C#</td>
<td>Visual Basic</td>
</tr>
<tr>
<td><strong>Projects</strong></td>
<td>Large, focus on reliability</td>
<td>GUI applications for windows</td>
<td>Web applications</td>
</tr>
</tbody>
</table>
General Results

- Conjecture was not confirmed
- Results of the study suggest
  - More efficiency while using create-set-call
  - Programmers preferred create-set-call

Reminder: conjecture stated that required constructors are superior to create-set-call
Task: File API Usage/Design

- Systematic programmers: design a file API
- Others: write code without IDE
  - Using file API the programmer would expect
  - Code should send contents of file by e-mail

Results

- All participants provided/used default constructors
- Designed APIs also contained required constructors
Task: Domain-Independent Classes

- Using objects in a setting with no prior intuition
  - Plausible, but not understandable class names
  - Task consisted of correctly initializing object and then calling `use()` function
- Compiler error vs. runtime exception

Results
- Many attempts to simply pass `null`
- Exceptions more useful than compile errors
Nearly all participants preferred create-set-call

- Initialization flexibility – independent of creation
- More control – error handling with return codes
- Consistency
- Less restrictive
Study Limitations

- Official limitations
  - Short tasks – advantage for default constructor?
  - Generalization to other programming languages?
- My opinion
  - Very few participants – representative?
  - Recruitment procedure justified?
Conclusion

- Study suggests that create-set-call
  - ... is expected by many programmers
  - ... provides greater readability and debugability
  - ... gives the programmer control and flexibility

→ Write your APIs using default constructors!