Software Engineering
Prof. Dr. Bertrand Meyer
Dr. Manuel Oriol
Dr. Bernd Schoeller

Lectures 25: Documentation
Today

- Documentation
  - Marketing
  - User Documentation
  - Design/Architectural Documentation
  - Technical Documentation
MARKETING DOCUMENTATION
Idea

- Have a catchy document that encourage people to know more about the product

- Give an idea of the functionalities

- Show how well it does it
Learn About Java Technology

To date, the Java platform has attracted more than 5 million software developers. It's used in every major industry segment and has a presence in a wide range of devices, computers, and networks.

Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

Java powers more than 4.5 billion devices including:

- 800 million PCs
- 1.5 billion mobile phones and other handheld devices (source: Ovum)
- 2.2 billion smart cards
- Set-top boxes, printers, Web cams, games, car navigation systems, lottery terminals, medical devices, parking payment stations, and more.

To see places of Java in Action in your daily life, explore java.com.

http://www.java.com/en/about/
Why Software Developers Choose Java

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

- Write software on one platform and run it on virtually any other platform
- Create programs to run within a Web browser and Web services
- Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
- Combine applications or services using the Java language to create highly customized applications or services
- Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

http://www.java.com/en/about/
Example: Eiffel Marketing (1/2)

http://www.eiffel.com/
**What developers at AXA Rosenberg say about Eiffel:**

“We can produce systems 3 to 4 times more efficiently with Eiffel than we could with C or Fortran.”

“Eiffel makes it dramatically easier to do what you say you do. In fact, using the Eiffel method allows you to do things you couldn’t do before. We view Eiffel as a language for simulating reality.”

“Eiffel has assertions built into the language, so you can actually specify in assertions exactly the behavior that you want.”

“We view Eiffel as a knowledge-sharing medium. Our application would have certainly been harder to do in any other programming languages, and it would have been impossible to do well.”

“EiffelStudio has opened up a whole new world for us by providing immediate and effortless migration to .NET. We can more easily incorporate third-party software modules and source code than we could with COM objects.”

How to present it?

- Target your audience well enough
- Do not hesitate to oversimplify if needed
- Emphasize your main points using graphs and tables
**Comparative tables**

**Blog software comparison chart**

Updated May 18, 2006: This chart is a companion to the article "Time to check: Are you using the right blogging to note that this chart reflects only standard features of the tools' installation, and not every possible extension, plug-in.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Blogger</th>
<th>TypePad Basic</th>
<th>TypePad Plus</th>
<th>TypePad Pro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comments</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Categories</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Subcategories</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Trackbacks</td>
<td>Yes (Backlinks)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pings</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RSS</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Atom</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Search</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Blogroll/Listes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Number of blogs</td>
<td>Unlimited</td>
<td>1</td>
<td>3</td>
<td>Unlimited</td>
</tr>
<tr>
<td>News Aggregation</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Extras</td>
<td>Blogger</td>
<td>TypePad Basic</td>
<td>TypePad Plus</td>
<td>TypePad Pro</td>
</tr>
<tr>
<td>Moblogging</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Photo Galleries</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Non-blog pages</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Blogger</td>
<td>TypePad Basic</td>
<td>TypePad Plus</td>
<td>TypePad Pro</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[http://www.ojr.org/ojr/images/blog_software_comparison.cfm](http://www.ojr.org/ojr/images/blog_software_comparison.cfm)
Comparison between IIS6 and Apache web servers. Data may be up to 1 hour old.

http://www.coderjournal.com/tags/iis/
Graphs

http://www.coderjournal.com/tags/iis/
USER DOCUMENTATION
Idea

- Users should be able to use the program. The documentation acts as a contract indicating what the program should do.

- It should be complete!

- Graphical user interfaces are not sufficient (even if everything is written down)
Types of documentation

- Thematic description
- Tutorial
- List of topics/references
Example of Thematic descriptions

- Encyclopedia
- Courses
- Textbooks
Tutorials

- A tutorial shows how to achieve a result with the software

- It guides users step by step

- It relies on people’s capability to extrapolate
**Example of list of ref.: Man pages**

**NAME**

`man` - format and display the on-line manual pages

**SYNOPSIS**

```
```

**DESCRIPTION**

`man` formats and displays the on-line manual pages. If you specify `section`, `man` only looks in that section of the manual. `name` is normally the name of the manual page, which is typically the name of a command, function, or file. However, if `name` contains a slash (/) then `man` interprets it as a file specification, so that you can do `man ./foo.5` or even `man /cd/foo/bar.1.gz`.

See below for a description of where `man` looks for the manual page files.

**OPTIONS**

- `-C config_file`
  Specify the configuration file to use; the default is `/private/etc/man.conf`. (See `man.conf(5)`.)

- `-M path`
  Specify the list of directories to search for `man` pages. Separate the directories with colons. An empty list is the same as not specifying `-M` at all. See SEARCH PATH FOR MANUAL PAGES.
Issues

- Most users don’t read documentation (RTFM) unless they have a problem...

- No matter what, documentation is dependent on the version of the software

- Searching through documentation can be a burden: it is the most difficult point
ARCHITECTURAL DOCUMENTATION
Idea

- Outlines the general structure of the software

- The goal is that people understand the general infrastructure

- It does not describe the implementation it describes the architecture and alternatives for the implementation if any

- Most of the time: figures with lengthy explanations
Example: Java Platform

http://www.sun.com/software/opensource/java/intro_java_tech.jsp
Example: Tomcat server

TECHNICAL DOCUMENTATION
Idea

- The documentation of the code itself

- Donald Knuth stated that the documentation should be written along the code as it is extremely difficult to write it afterwards

- Dependent on the language used

- Can be extracted automatically from the code
Example: Javadoc

This document is the API specification for the Java 2 Platform Standard Edition.

See:
- Description

### Java 2 Platform Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>java.applet</code></td>
<td>Provides the classes necessary in applet context.</td>
</tr>
<tr>
<td><code>java.awt</code></td>
<td>Contains all of the classes.</td>
</tr>
<tr>
<td><code>java.awt.color</code></td>
<td>Provides classes for color handling.</td>
</tr>
<tr>
<td><code>java.awt.datatransfer</code></td>
<td>Provides interfaces and classes related to drag and drop.</td>
</tr>
<tr>
<td><code>java.awt.dnd</code></td>
<td>Drag and Drop is a direct interface mechanism to transfer information in GUI.</td>
</tr>
<tr>
<td><code>java.awt.event</code></td>
<td>Provides interfaces and classes for handling events.</td>
</tr>
<tr>
<td><code>java.awt.font</code></td>
<td>Provides classes and interfaces for font handling.</td>
</tr>
</tbody>
</table>
Example: Eiffel Documentation
Example: Eiffel Documentation Formats
Conclusions

- Documenting software is NOT an easy task

- Most things can be presented in several way: the good way is the one that bears best the message intended

- The type of documentation matters as it has several standard ways of being made
Today

- Documentation
  - Marketing
  - User Documentation
  - Design/Architectural Documentation
  - Technical Documentation