

Documenting APIs: Your First Week on the Job

Silicon Valley Chapter-STC

July 27, 2006

Jim Bisso, Bitzone LLC

jbisso@bitzone.com, 510.234.4046

Copyright © 2006 Bitzone LLC. All rights reserved.

Introduction

- **Jim Bisso**
 - technical communicator since 1988
 - specializing in developer documentation
 - co-author of *Documenting APIs: Writing Developer Documentation for Java APIs and SDKs*
 - instructor
 - computer science
 - technical writing
 - software engineer at Sun Microsystems

Overview

- Skills sets
 - skills versus procedures
 - what you bring
- Documentation
 - what you produce
- Resources
 - what you find

Copyright © 2006 Bitzone LLC. All rights reserved.

Often tech writers are tasked with documenting APIs. They may be doing other kind of documentation. No matter how they arrive at dev docs.

This is a framework for how I go about documenting APIs.

Skill Sets

- More a set of skills than a list of procedures
 - problem-solving skills
 - heuristics, described in 1945 in Pólya's *How to Solve It*
 - programming language skills
 - reading / writing
 - OOP concepts, OOA/OOD
 - domain knowledge (see resources)

Copyright © 2006 Bitzone LLC. All rights reserved.

The difference between reading a foreign language and writing one. Basic spoken phrases and writing poetry.

The ability to

- recognize class declarations or method signatures
- format and documenting code snippets
- write complete tutorials or sample applications.

Sets and lists: order and repetition.

The scientific / structural / logical methods. Top-down analysis, bottom-up synthesis. Messing around with systems. Play. Taken apart toys, clocks, cars, etc.

OOP concepts: identity, classification, inheritance, polymorphism, and persistence.

George Pólya. 1945. *How to Solve It*. (expand to its own slide?)

Understand the problem.

Make a plan.

Execute the plan.

Evaluate the solution. (How could it be better?)

Heuristics

- Pertaining to how something is discovered.
Polya described the steps in discovery as:
 - understanding the problem
 - making a plan
 - executing the plan
 - evaluating the solution
 - looking back
 - how could it be better?

Copyright © 2006 Bitzone LLC. All rights reserved.

Has somebody solved this problem before. If so, piggy back off their solution.

Top-down vs bottom-up analysis. And synthesis.

Working in parallel and managing tasks.

Getting feedback from your audience.

Developer Documentation

- Product / API
- Documentation
 - reference guide
 - programmer's guide
- Audience

Copyright © 2006 Bitzone LLC. All rights reserved.

What is the product? How does it work? What does it provide for the customer?

What kind of API is it? Plug-in vs (Class) Library. Example: Flash plug-in for web browser.

Documentation:

Reference guides. The structure is rather well-known on account of near-standard tools like Javadoc or Doxygen. A doc plan is not so much concerned with the structure of the reference guide or its contents as to the stability of the API, locating / rallying resources (human and written), and priorities). Tend towards a complete coverage of the existing classes and their constituent members.

Programmer's guides. Here there are fewer de facto standards and more of a need for a traditional doc plan. More how and why to use the API in the manner exemplified by the sample code.

Audience: Developers. Cf. Other, similar third-party APIs; review their doc set. For the reference guide they want their specific questions answered. Whether that is online (doc sets, customer forums, mail lists) or shipped with is of little importance.

Developer Doc Planning

- Coverage
 - packages (or namespaces), classes, members
- Milestones
 - coordination with product
- Deliverables
 - paper
 - help
 - web

Copyright © 2006 Bitzone LLC. All rights reserved.

Coverage. Document what's public (and sometimes what's protected).

Milestones. Reference guide can be kept in sync with product and have a near simultaneous release. APIs should be changing radically in the final period of development. Programmer's guides and/or sample applications (tutorials) are more difficult, and may be staggered soon after. Part of the importance of keeping the product web site fresh.

Deliverables. Fewer books than HTML. Whether delivered online / web or as part of the SDK / product set.

Resources

- Product
- Subject matter experts (SMEs)
- Domain knowledge

Product

- Get to know your product
 - earlier versions
 - documentation
 - functional specifications
 - existing manuals
 - competing products

Copyright © 2006 Bitzone LLC. All rights reserved.

The deltas / diffs between an earlier version and the current one need to be managed. Sometimes running the auto-gen tool on the current code gives some view of what's lacking, but not of which comment content is incorrect. Support and QA can be data minded, too.

Functional specs. If they don't exist, try to construct a high-level overview of the product and its sub-systems by interviewing the architect, product manager, or some senior programmers.

If this is not a unique product see what the competition is doing. Third-party books on foundational or category-specific domains of knowledge.

SMEs

- High level overview
 - architect
 - product manager, marketing & technical
- Details
 - engineering, deltas from functional spec
- Auxiliary
 - QA, testing plans
 - support: FAQs, forums

Copyright © 2006 Bitzone LLC. All rights reserved.

Get to know the team early on. Social engineering. Old timers and newbies. Where does tech pubs fit in? Smelling fear.

Granularity. The big picture and in the trenches.

A different view of things.

Domain Knowledge

- Foundational
 - programming language(s)
 - technologies
- Transferable
- Category-generic
- Implementation-specific

Copyright © 2006 Bitzone LLC. All rights reserved.

Levels of domain knowledge:

- Foundational
 - Programming language(s): Different kinds: OOP, procedural, functional, declarative. C, C++, C#, Java, Visual Basic, JavaScript, Perl, Ruby; Smalltalk, Lisp; Prolog.
 - Protocols, standards (e.g., XML): The XML family (XML, XSL, XSLT, XSL-FO, Schema / DTDs). The Web (Web services, SOAP, ASP, JSP, Struts, JSF, AJAX).
 - Design patterns and Anti-Patterns. Small design patterns (Gang of Four), System architectural design patterns. Wikis and extreme programming, networking, see C2.com.
- Transferable: from non-computer domains, e.g., accounting, writing, music
- Category-generic: CORBA, RDBMS, SOA (service-oriented architecture), REST (representational state transfer), types of software
- Implementation-specific: proprietary knowledge / technology of companies

Q & A

- Questions and comments