

# SDK

# PLANNING AND DEVELOPMENT

Jim Causey

Senior Programming Writer

Microsoft



# INTRODUCTION

- Thanks for coming out!
- Background and assumptions
- Survey
- Practical focus
- Not just about Microsoft technologies



# AGENDA

- Components of a successful SDK
  - Setting the stage
- Planning
  - Failing to plan is planning to fail...
- Scenarios
  - From “let’s change the world” to “what to do if you’re screwed”
- Execution
  - Tips, tricks, best practices
- Q&A
- Appendix: getting started

DEFINITIONS AND FEATURES

# **COMPONENTS OF AN SDK**

# SDK: DEFINITIONS

- Software Development Kit
- Source and samples
- Tools
  - Compilers
  - Build systems
  - Debuggers
  - QA
- **Docs!**

REVIEW

## Windows Mobile 6.5 Review: There's No Excuse For This

By [John Herrman](#), 3:01 AM on Tue Oct 6 2009, 60,903 views

56 diggs [digg it](#)



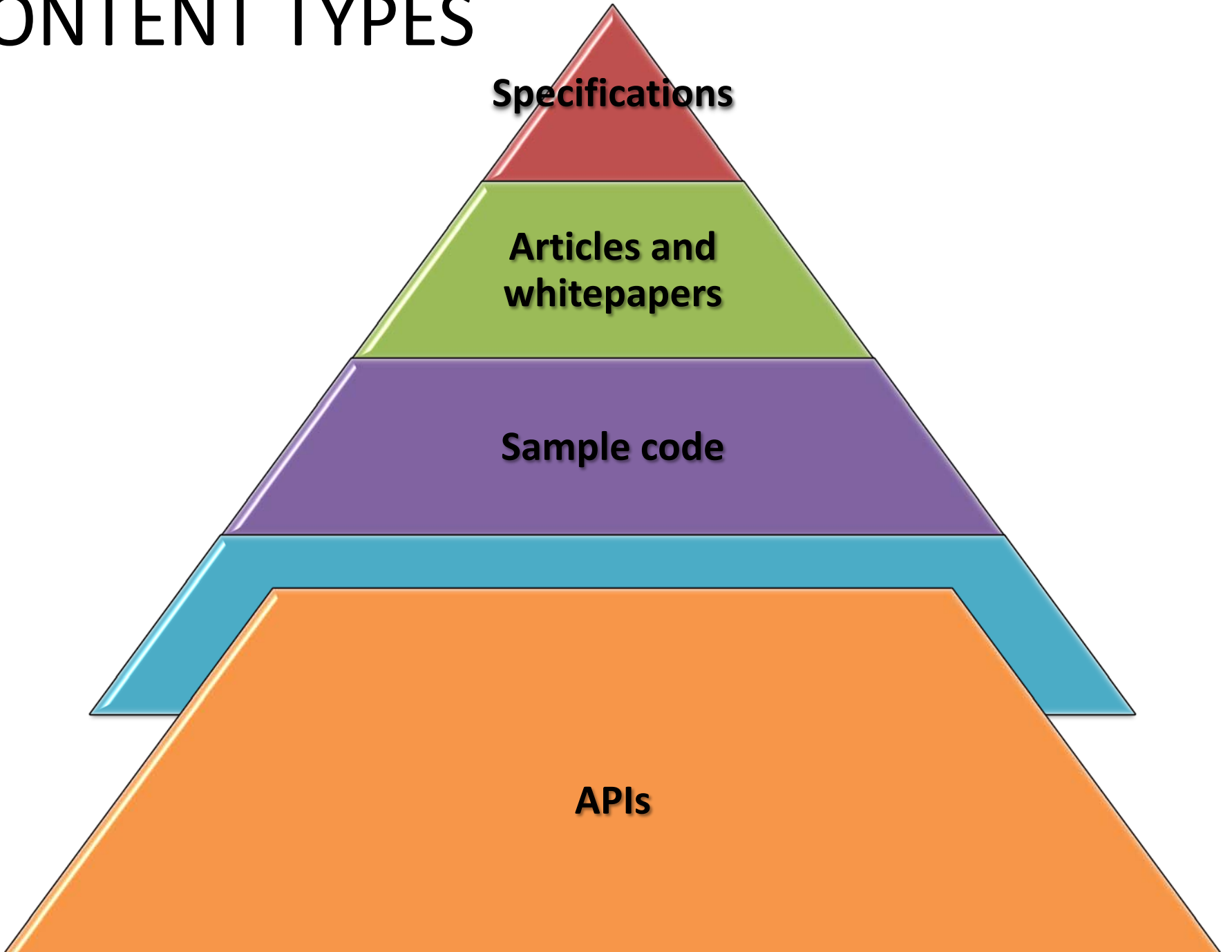
# CONTENT TYPES

**Specifications**

**Articles and  
whitepapers**

**Sample code**

**APIs**



# APPLICATION PROGRAMMING INTERFACES (APIs)

- **Bread-and-butter of the role**
- Detailed programming element documentation
  - Functions
  - Classes
  - Structures
  - Predefined variables
- Combination of automated tools and manual writing



# API TYPES

- Assembly language
  - Symbolic representation of hardware-specific instructions
  - Common languages: x86, ARM, MIPS, Power PC
- Native code
  - High-level source code compiled to machine code
  - Common languages: C/C++
- Managed code
  - High-level source compiled to virtual machine bytecode
  - Common languages: Java, C#, VB.NET, UCSD Pascal
- Interpreted code
  - High-level source executed step-by-step by an interpreter program
  - Common languages: BASIC, Perl, PHP

# CONCEPTUAL DOCUMENTATION

- Big-picture guidance on API usage
  - Overviews
  - How-to
  - Descriptions
  - Best practices
- Writers **must** be able to understand specifications and source code
- Keep international audiences in mind!

# SAMPLE CODE

- Snippets
  - Demonstrate syntax and usage
  - Liberally scattered through other doc types
- Samples
  - Complete bundles of code
  - Demonstrate practices end-to-end

# GUIDES AND WHITEPAPERS

- Standalone guidance on a major feature or set of features
- End-to-end coverage
- **Shaping the story for the product**
- Often includes all of the previous types of developer docs along with some sample code

# TECHNICAL SPECIFICATIONS

- Low-level details on implementation of a platform
- Used by both hardware and software engineers
- Often face specific legal requirements

HOW TO DECIDE WHAT TO DO

# **PLANNING CONSIDERATIONS**

# PROJECT TRIANGLE





# ELEMENTS OF AN SDK PROJECT

## VISION

- Audience
- Goals & non-goals
- Guiding principles
- Definition of success

## DETAILS

- Deliverables
- Completeness
- Tone
- Process

## SCHEDULE

- Milestones
  - Planning
  - Execution
  - QA
  - Release
- Iteration
  - Review
  - Adjust

# BUILDING YOUR VISION DOCUMENT

STAKEHOLDERS

AUDIENCE

RESOURCES

GOALS & NON-  
GOALS

VISION



# STAKEHOLDERS

- OARPI model
  - Owners
    - Who is driving the plan?
    - Butt-on-the-line (BOL)
  - Approvers
    - Go/no-go decisions
    - Project review
  - Reviewers
    - Critical dependencies
    - Technical and project reviewers
  - Participants
    - Contributors
    - Feature requestors
  - Informed



# AUDIENCES

- Audience types
  - Software engineers
  - Hardware engineers
  - Students
  - Hobbyists
- Variations
  - Established markets
  - New products
  - Key influencers

# HUMAN RESOURCES

- Writers
  - Full-time vs leverage
- Editors
- Product team
  - Developers
  - Testers
  - Program managers
  - Planning and marketing
- Evangelism
- Product support
- Legal
- Customers and community

# TECHNICAL RESOURCES

- Managed vs native code
- Open-source vs closed-source
- Tools and automation
  - Authoring
  - QA
  - Documentation build
  - Source code comments
- Product type (local vs web)
- SDK destination (local install vs web)

# GOALS & NON-GOALS

- What we're doing and not doing
- Specific
- Actionable
- Measureable
- Example goals:
  - Provide a reference topic for every public API by project end
  - Develop command-line help for the top three tools in the SDK
- Example non-goal:
  - We will not provide code samples

DISCARD YOUR ASSUMPTIONS

# **PLANNING IMPLICATIONS**

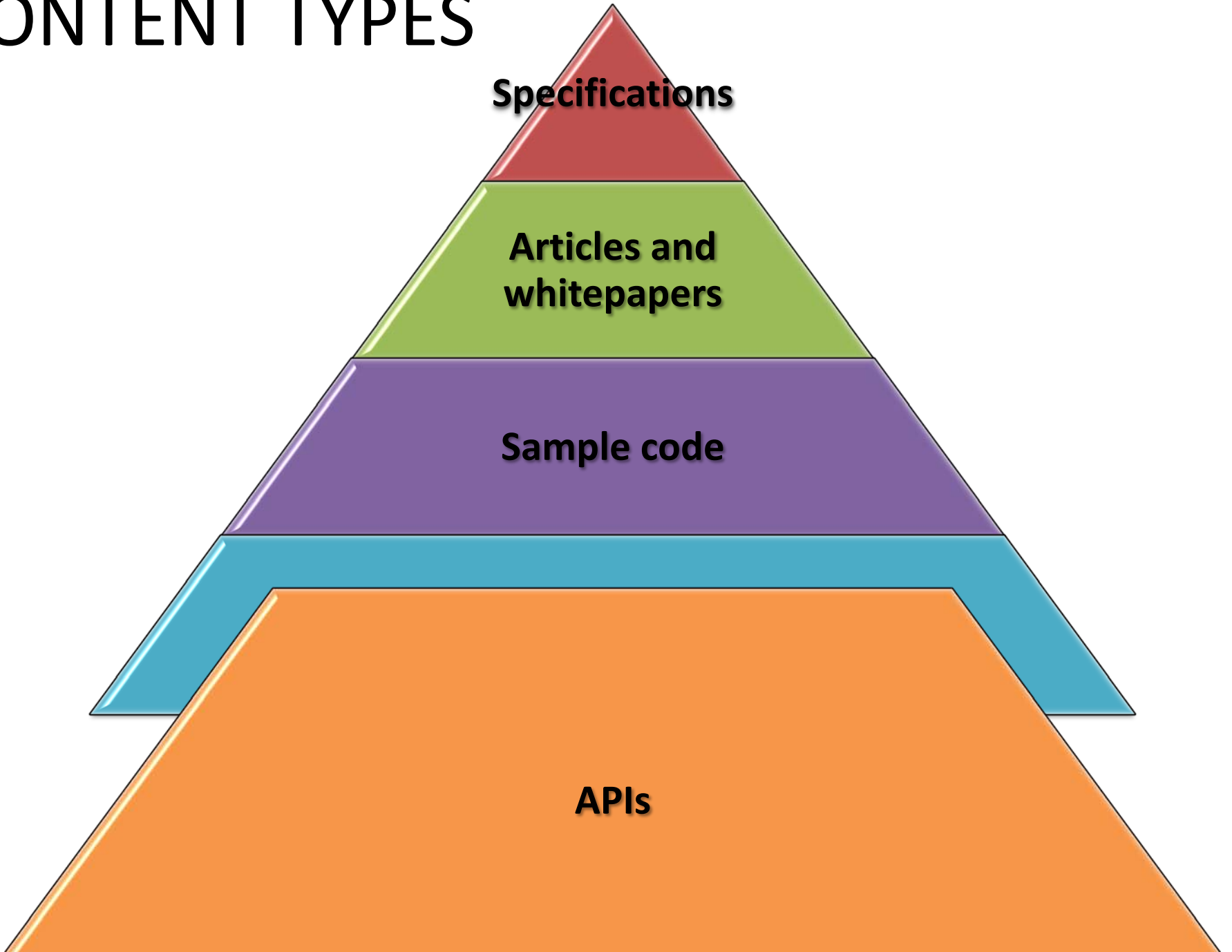
# CONTENT TYPES

**Specifications**

**Articles and  
whitepapers**

**Sample code**

**APIs**





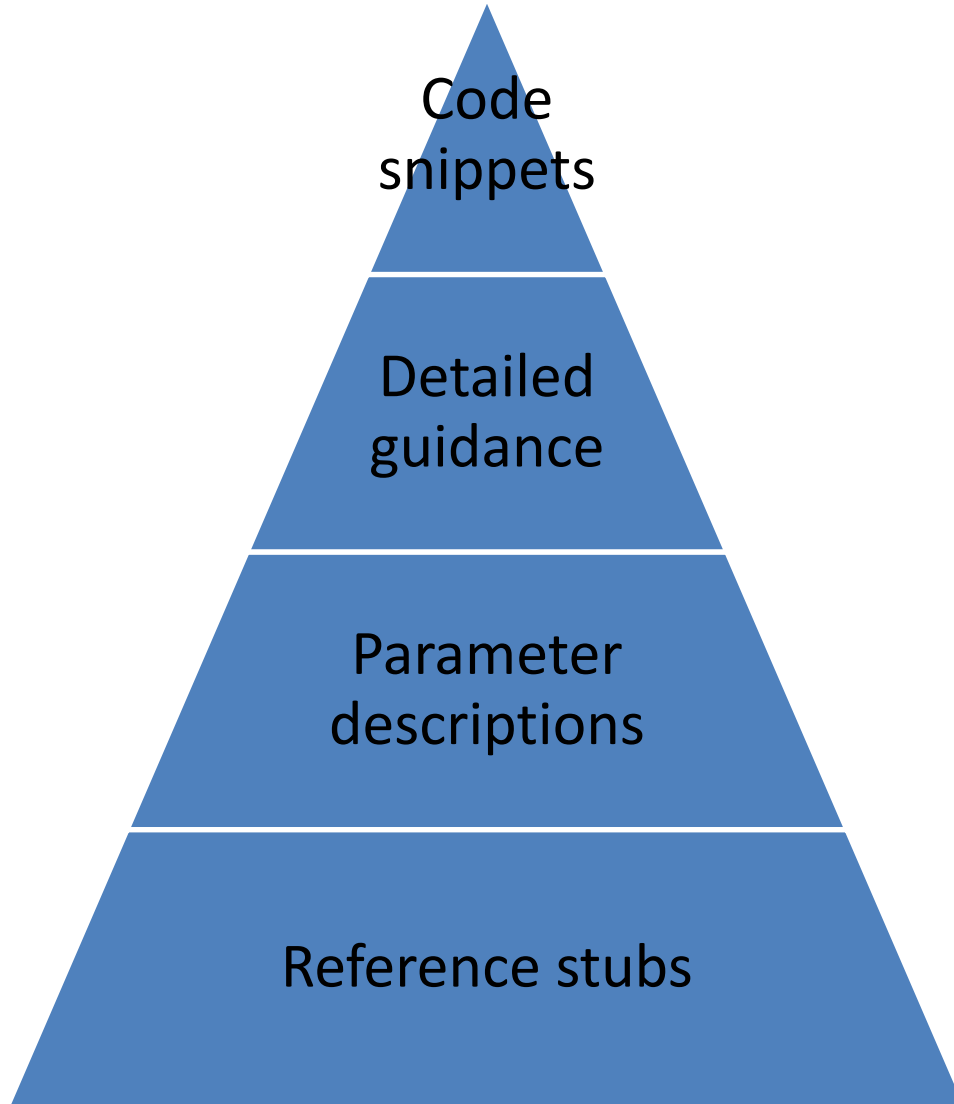
# WHAT IS SUCCESS?

- The trap of completeness
- Support costs
- Legal constraints
- Product adoption
- Customer reviews

# REFERENCE PLANNING

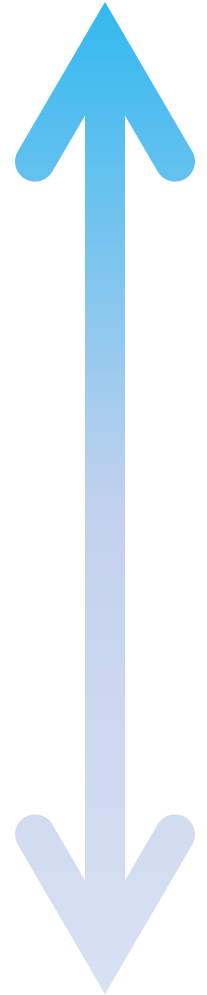
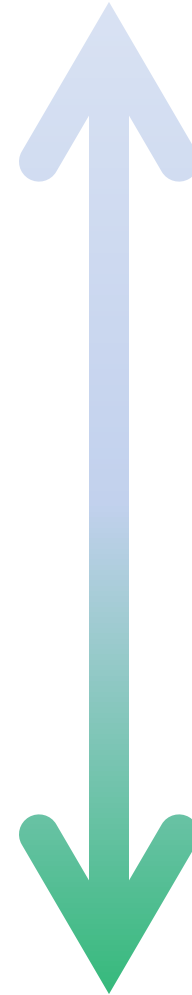
- Reference is the normal bottom line
  - What about open-source code?
  - What about automated docs?
    - With and without code comments
    - JAVADOC and Sandcastle
  - What about object reflection?
  - What about legal requirements?

# REFERENCE QUALITY VARIANCE



Automation

Technical Skill



# CONCEPTUAL PLANNING

- Bringing the story together
- Requires more time and expertise than reference
- Requires less expertise than code samples
- What if you only shipped conceptual docs?
- What if you don't ship any conceptual docs?
- What if you rely on the product team for conceptual docs?

# SAMPLE PLANNING

- Pros
  - Developers love sample code
  - Can stand alone
  - Quality vs quantity
- Cons
  - Expensive
  - Requires technical expertise or engineering commitment
- Consider having only samples
  - Use same standards as product code
- Consider having no samples
  - Limited resources, driven by legal requirements
- Consider using only snippets or pseudocode
- Consider leveraging samples
- Refer to Leah's presentation for more guidance

# SCENARIOS

# CHANGING THE WORLD

- **Goal:** Build a new developer platform to compete with an industry leader
- **Resources:**
  - Programming writers with editors
  - Content build team
  - Managed language
  - Complete SDK
  - Integrated with engineering
- **Vision:**
  - Deep architectural and value-prop overviews
  - Complete walkthroughs and sample code for top 50 scenarios
  - Manually edited reference for each API member
  - Code snippets for 500 most common classes
  - All sample code compiles and builds with no security errors
  - Help and reference for all tools
  - Evangelist deep-dives on blogs
  - Public wiki
  - Active participation on StackOverflow.com

# MAINTENANCE

- **Goals:** support the release of a dominant platform; compatibility and legal concerns
- **Resources:**
  - Technical writers and some programming writers
  - Some engineering integration
  - Native code
- **Vision:**
  - Members and parameters for every reference topic
  - Fix top 300 customer bugs
  - Snippets and complete descriptions for top 1000 topics
  - All sample code is free of security bugs
  - 30% of sample code compiles
  - 10% of sample code passes all automated quality tests

# TIGER TEAM

- **Goals:** innovative product for an advanced audience needs to get more with less
- **Resources:**
  - No dedicated writers
  - Leverage .NET tools and features
- **Vision:**
  - Rely on object reflection for all reference
  - One detailed end-to-end sample application
  - Comic book describing architecture and goals
  - Blog articles providing scenario guidance

# OH, SHIT

- **Goals:** doc project launched late in cycle with technical writers
- **Resources:**
  - No content build and tools
  - Technical (not programming) writers
  - Engaged dev/test/PM
- **Vision:**
  - Enlist program management to draft architecture guidance
  - Rely on testers for all sample code
  - Use code comments and automation for all APIs
  - Technical writers convert specifications into limited conceptual docs



DO AS I SAY, MAYBE NOT AS I DO...

# **BEST PRACTICES**

# SCHEDULING

- Dedicate time to planning
- Dedicate time to technical review
- Break project into milestones
- Align milestones with product team
- Revisit plan and schedule at each milestone
- Committed/Targeted/Cut

# PROJECT REVIEW

- Schedule project reviews early and often
  - Beginning of project and during each planning sub-milestone
- Involve senior leadership
  - Signoff from engineering (dev/test/PM)
  - Signoff/review from other stakeholders
- Iterate and revise

# ENGINEERING INTEGRATION

- Know your stuff
  - Understand the competition
  - Understand the product
- Don't waste people's time
  - Dig into specs and source before interviewing
  - Work through drafts and ask targeted questions
  - Ask smart, incisive questions
- Meet or exceed engineering standards
  - Scheduling
  - Deadlines
  - Quality
- Make tough cuts

# COMPETITIVE ANALYSIS

- Read lots of documentation
- Work through how-tos and samples
- Steal best practices
- Understand pitfalls

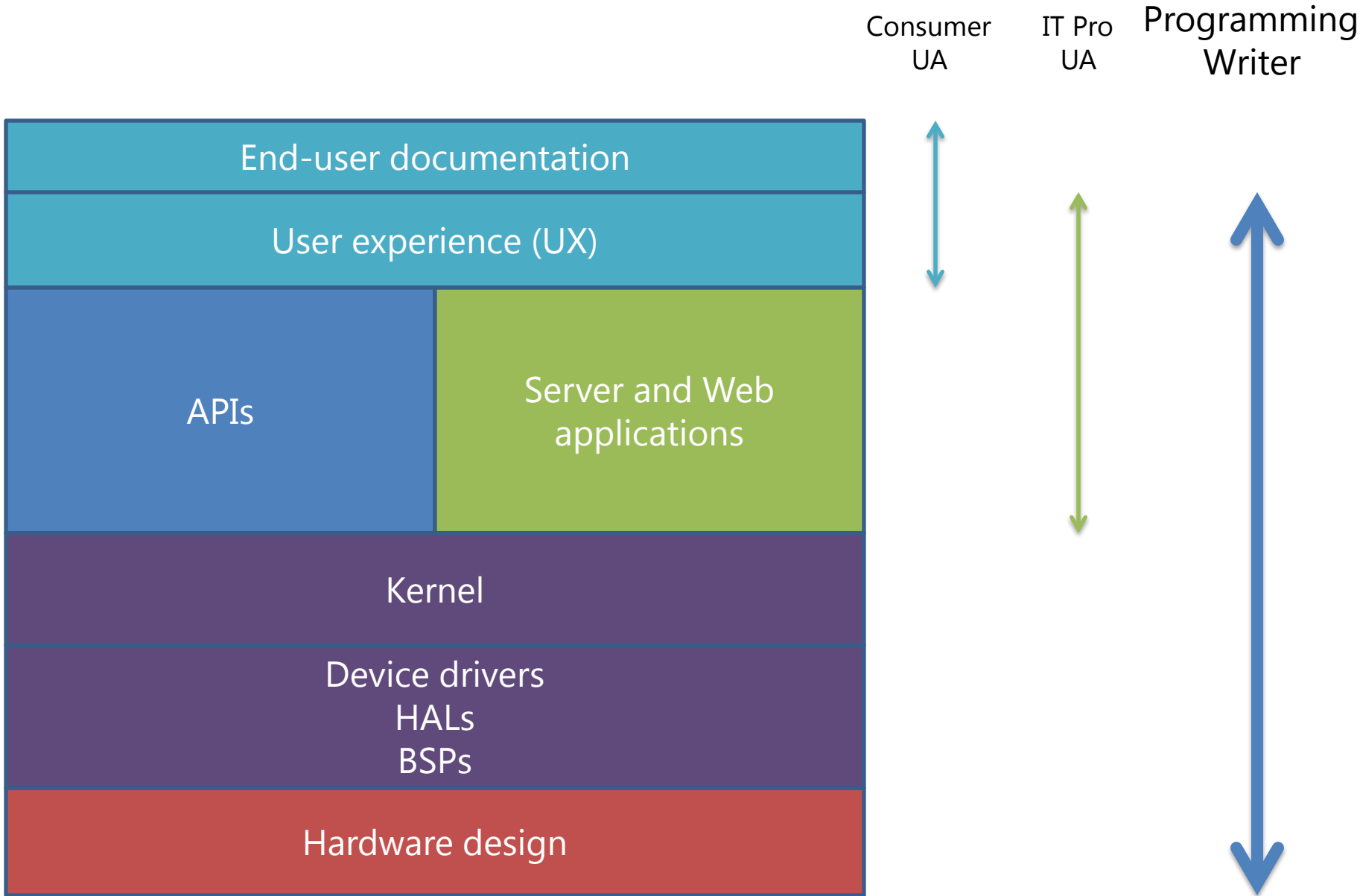
# EMPOWERMENT

- Own your documentation
  - Don't defer to dev/test/PM
  - Don't defer to marketing/support
  - Partner, but drive
  - Decide what's right and build your plan
- Make tough cuts
- Occupy the space

APPENDIX ONE

# **BECOMING A PROGRAMMING WRITER**

# UA PRODUCT SCOPE



# ENGINEERING SCOPE

- 1000 ft. view of product design, development, and features
- Peer relationships
  - Software developers
  - Software testers
  - Program managers
  - Architects
  - UX designers
  - Legal
  - Marketing
  - Product support

# CAREER PATHS

- Programming writer
  - Seniority as individual contributor
  - UA management
- IT pro writer
- Program management (PM)
- Development and test
- Developer support
- Developer evangelism
- Consultant

How do I become a programming writer?

# **WORK SKILLS**

# TECHNICAL FUNDAMENTALS

- Formal computer science background **not** required
- Passion and knowledge **are** required

Basic skills and terminology

Operating system fundamentals

Basics of compilers, interpreters

Runtimes and type systems

Memory management and garbage collection

Structured markup (XML and HTML)

# PROGRAMMING LANGUAGES

- The ability to **read** and **write** source code are critical
  - You don't have to be a world-class programmer... but you should **enjoy** coding

Scripting languages	Managed languages	Native code	Functional languages	Low-level
Perl	Java	C	LISP	68K assembly
Python	C#	C++	Scheme	x86 assembly
Windows PowerShell	VB.NET		Ruby	ARM assembly

# EVERYDAY WORK SKILLS

- Authoring systems
  - XML and structured markup
  - HTML and CSS
  - Office and DTP applications
- Engineering systems
  - Bugs
  - Code
  - Builds
- Project management

# BONUS TECHNICAL SKILLS

- Comfort with web publishing technologies
  - Static content (HTML/CSS)
  - Dynamic content (ASP.NET, PHP, Ruby on Rails)
  - Web servers (IIS, Apache)
- Comfort with DBMS
  - Expertise with SQL
- Experience with systems administration
- Familiarity with multiple technical support tiers
- One or more rounds shipping commercial software products

# SOFT SKILLS

- **Programming writers don't sit in their offices and wait for work**
- CREDIBILITY
  - Interview senior engineers at their level
  - Own a room filled with technical experts
  - Do your research and homework
  - Minimize randomization
- POSSESSIVENESS
  - **Own your documentation**, and the story of your product
  - Don't work like a service bureau
  - Close gaps, get things done
- DETERMINATION
  - Work all the technical and business channels to understand your features
  - Use every trick to get technical reviews

# GETTING STARTED

- Get started coding!
  - Use classics of the genre – learn coding and great programming writer style!
    - **Consider best and worst doc examples as you go!**
    - See Appendix I
  - Target a market and learn the associated language
    - .NET/Java (managed languages)
    - Operating systems (C/C++/assembly)
    - Scripting and web development (Ruby/PHP/Perl/Python/.NET)
  - Write real apps
    - Start with simple “Hello, World”
    - Work up to something you’re passionate about
- Get started documenting code!
  - Documentation for an open source project
  - Write a sample to fill in the gaps on some bad docs
  - Blog about a coding problem you’ve solved

APPENDIX TWO

# **SAMPLES AND REFERENCES**

# CLASSICS BY PROGRAMMING WRITERS

- *Programming Windows* by Charles Petzold
- *Code: The Hidden Language of Computer Hardware and Software* by Charles Petzold
- *Windows Internals* by Mark Russinovich, David Solomon, Alex Ionescu
- *Mac OS X Internals: A Systems Approach* by Amit Singh
- *The Pragmatic Programmer: From Journeyman to Master* by Andrew Hunt, David Thomas
- *The C Programming Language* by Brian W. Kernighan, Dennis M. Ritchie
- *Code Complete: A Practical Handbook of Software Construction* by Steve McConnell
- *Programming Perl* by Larry Wall, Tom Christiansen, Jon Orwant

# DOC SAMPLES

- **API documentation**

- Native code
  - Win32 API: *CreateWindow()*
    - [http://msdn.microsoft.com/en-us/library/ms632679\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms632679(VS.85).aspx)
  - iPhone API: *audioPlayerDidFinishPlaying:successfully:*
    - [http://developer.apple.com/iphone/library/documentation/AVFoundation/Reference/AVAudioPlayerDelegateProtocolReference/Reference/Reference.html#//apple\\_ref/occ/intfm/AVAudioPlayerDelegate/audioPlayerDidFinishPlaying:successfully:](http://developer.apple.com/iphone/library/documentation/AVFoundation/Reference/AVAudioPlayerDelegateProtocolReference/Reference/Reference.html#//apple_ref/occ/intfm/AVAudioPlayerDelegate/audioPlayerDidFinishPlaying:successfully:)
- Managed code
  - Silverlight *Panel* class:
    - [http://msdn.microsoft.com/en-us/library/system.windows.controls.panel\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/system.windows.controls.panel(VS.95).aspx)
  - *Java.applet* package
    - <http://java.sun.com/j2se/1.5.0/docs/api/java/applet/package-summary.html>

- **Conceptual documentation**

- Win32: About Window Classes
  - [http://msdn.microsoft.com/en-us/library/ms633574\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms633574(VS.85).aspx)
- Silverlight layout system
  - [http://msdn.microsoft.com/en-us/library/cc645025\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc645025(VS.95).aspx)
- iPhone AVAudioPlayer Class Reference
  - <http://developer.apple.com/iphone/library/documentation/AVFoundation/Reference/AVAudioPlayerClassReference/Reference/Reference.html>

# DOC SAMPLES (CONTINUED)

- **Sample Code**
  - Serial Communications in Win32:
    - <http://msdn.microsoft.com/en-us/library/ms810467.aspx>
- **Guides and Whitepapers**
  - Application Platform Overview for Windows Phone
    - <http://go.microsoft.com/?linkid=9713249>
  - Windows Phone UI Design and Interaction Guide
    - <http://go.microsoft.com/?linkid=9713252>
  - The Java Servlet API White Paper
    - <http://java.sun.com/products/servlet/whitepaper.html>
  - Mac OS X Technology Overview
    - [http://developer.apple.com/mac/library/documentation/MacOSX/Conceptual/OSX\\_Technology\\_Overview/About/About.html](http://developer.apple.com/mac/library/documentation/MacOSX/Conceptual/OSX_Technology_Overview/About/About.html)
- **Technical Specifications**
  - Microsoft Open Specifications:
    - <http://go.microsoft.com/fwlink/?LinkId=111240>
  - W3C Standards:
    - <http://www.w3.org/standards/>