# Nitin's Thumb Rules for Common Sense Software Development

By Nitin Bhide
*(nitinbhide@yahoo.com)*

Jan 2004

# The Thumb Rules

- Observe and Understand your customer
- Make it Goof Proof
- First, Do no Harm
- Add Great Extras
- Beware of Feature Creep

# The Thumb Rules …

- One Click is Simple than Two.
  - Keep it Simple (Simple as Frisbee)
- Have a Backstage Pass
- Common Practice is NOT ALWAYS Common Sense
- The Passion Factor - *I will not Ship Shit*

# Observe and Understand your customer

- Find out who is your customer.
  - If you are developing a product, End User is your customer
  - If you are developing a library, Application Developer is your customer

# Observe and Understand your customer

- Understand the your customers needs *(and not just expectations)*
  - *Customer may have unreasonable expectations but his needs are real.*
  - *Needs and expectations can be totally different.*
  - *Once you understand the need, you can give satisfactory solutions*

# Observe and Understand your customer

- Observe how customer is using an existing application
  - What are the current pain areas ?
    - Is Performance a problem ?
    - Is user interface /API a problem (cryptic, confusing, too many alternatives)?
    - Is he trying to fit rectangular peg in round hole ?

# Make it Goof Proof

- Remember that your Customer is Human.
  - Hence he is bound to make mistakes at times.
  - If you make sure that mistakes are caught early, you are making him productive.
- Don’t violate commonly accepted conventions.
  - E.g.. Don't put File Button at the end of Menu bar.

# Examples of ‘Make it Goof Proof’

- Application Developers
  - Undo,
  - Error messages/recovery,
  - AutoSave features
- Library Developers :
  - Use assertions
  - Give attention to error handling
  - Provide logging and support for easier debugging

# First, Do No Harm

- Doctors principle applied to Software
- Take the pain/struggle/frustration out of your product, you will WIN over the customer.
- Ensure that customer doesn't loose his data under any circumstances.

# Examples of ‘First, Do No Harm’

- Clearly separate Inputs and Outputs for functions.
- Avoid using same variable for input and output
- If using same variable, do not change it unless your function is successful.
- In C++, Use ‘const functions’ and ‘const variables’.

# Add Great Extras

- At times, ‘Adding a Great Extra’ may make or break a product.
  - The Extra can be someone else’s product added to your mix.
  - The Extra can be something you have developed.

# Examples of ‘Great Extras’

- Visual C++ is popular not because the Compiler is best, but because IDE easier to use.
- Support for multiple CAD file formats for CAD Application.
- Add Support for writing *Addins* to the application
- Giving a Installer/Uninstaller rather than ‘installation instructions sheet’.

# Examples of ‘Great Extras’

- Support to multiple compilers to your library
- Using a third party library (Geometry Kernel, Standard Template Library etc).
  - E.g. ‘This library is available on ACIS or Parasolid’ – Great. Customer’s efforts/cost/development time are reduced.
- Adding a Debugging support.
- Making Sources available (for free or for fee)
  - Source availability is a ‘Great Extra’ for all Open Source software

# Beware of Feature Creep

- Adding unnecessary bells & whistles complicates software
  - It will make it difficult to design and implement.
- If your users are struggling with manual trying to find how to do a common task, something is wrong with the product.
- Simplicity is Essential.

# Beware of Feature Creep

- Challenge every feature.
  - For every feature, ask ‘why?, why?, why?’
- Chuck out preconceptions of ‘*how complex’* something needs to be ?

# Beware of Feature Creep

- Spend more time driving your own product.
  - Try ‘driving’ them as if you are trying them for the *first time*.
  - Understand which features are ‘*necessary*’ and which are not
  - You are having problems using the software definitely your users will have problems
  - If you don’t like Unit Testing your product, *mostly likely your users won’t like to test it either*

# One click is Simple Than Two

- Make your product Faster and Simpler to use then it has a better chance of success.
- Don’t let Cloud of features Blur the *most common use* of your product.
- Target for average common user and have advanced features available for experts.

# One click is Simple Than Two

- Strive for SIMPLICITY
  - In Architecture Design,
  - In Coding
  - In API Design
  - In User Interface
  - In Documentation

# Examples – *One Click is Simple than Two*

- Amazon.com – Its simple one click ordering process is a major of factor to make is popular
- Pop Up /Context Sensitive Menu – Most needed tasks are available on single click on ‘third button’.
- Drag/Drop File Open

# Examples – *One Click is Simpler than Two*

- Visual Installers rather than install scripts/documents.
  - InstallShield and Wise created a Industry around a simple user need.
- Give a simple basic API and separately give advanced API.
- Maintain the consistency of API.

# Have a Backstage Pass

- Give your customer a view of what is happening behind the scene.
- This gives them a sense of security.
- Also provides them a view of what will work and what won’t work.

# Examples ‘Have a Backstage Pass’

- Facility to provide detailed logs from the application or library
- Details of implemented/used algorithms
- Document format of the support files
- A Progress Bar is a classic example of ‘a backstage pass’

# Common Practice is NOT ALWAYS Common Sense

- Yesterdays *Common Sense* typically becomes today's *Common Practice*
- Common Practices doesn't necessarily mean its Common Sense in Today's circumstances and context
- Following Common Practice WITHOUT Understanding the Context is a recipe for disaster

# Classic Example of Common Practice and Common Sense

- ‘For Portable code, Don’t use templates or STL’
  - Common Sense 5 years back
  - Common Practice Today (not a Common Sense Any More)

# Classic Example of Common Practice and Common Sense

- Circumstances 5 years back.
  - C++ Compiler on most of Unix platforms did not support templates or STL
  - Templates and STL are NOT part of C++ standard.
  - Inconsistencies between different implementations of templates and STL
- Hence, *Don't Use STL* is a Common Sense 5 years back.

# Classic Example of Common Practice and Common Sense

- Circumstances Today
  - Templates and STL are Part of C++ Standard
  - Almost all compilers support templates and STL
  - Still few inconsistencies in template implementation on different platforms
- Todays Common Sense – *Use STL and Carefully use Templates*

# Other Examples of 'Common Practices'

- "Starting a new Project ?"
  - "Use COM"
  - "Use ActiveX"
  - "Use Java"
  - "Use XML"
  - "Use Web Services"
  - "Use RUP"
  - "Use UML"

# The Passion Factor

- All the above Thumb rules are useless if there is NO PASSION.
- Passion Fuels the *Fire*
- 'Passion for work', 'Passion for Quality' will make sure that all other Thumb Rules happen.

# The Passion Factor

- For A Passionate Team, Nothing is Impossible.
- Passionate Teams need ‘a tangible goal’. *Something to shoot for and Be Proud Of.*
- Passionate Teams need ‘Recognition’ of the efforts to sustain and fuel the ‘passion’

# The Passion Factor

- Only a Passionate Leader can Nurture and Sustain a Passionate Team

# References


- Art of Innovation – Tom Kelly