All we like sheep: Cloning as an engineering tool

Michael W. Godfrey
University of Waterloo

Quotes on source code cloning

"Q: You want to stop developers cutting and pasting code? Why?

A: This is Software Engineering 101, for heaven's sake!!"

– http://www.anticutandpaste.com/

Quotes on source code cloning

"So, copy-and-paste is not necessarily bad in the short run, if you are copying good code. But it is always bad in the long run."

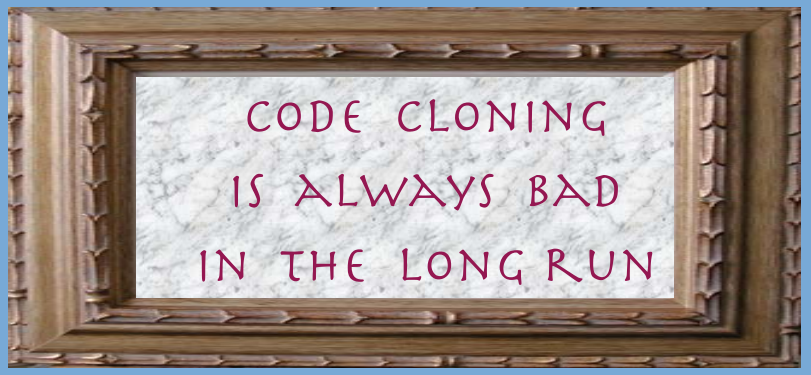
– Ralph Johnson, 2004 blog [1]

Quotes on source code cloning

"Number one in the stink parade is duplicated code. If you see the same code structure in more than one place, you can be sure that your program will be better if you find a way to unify them."

– "Bad Smells" [Beck/Fowler in *Refactoring*]

Myth



Why cloning is supposed to be bad

- Duplicated code leads to bloat
  - Hard to understand, less “essential”
  - Will all bug instances be fixed?
- Duplication is a sign of inexperienced developers
  - Copy/paste is often “easy”, JIT comprehension
  - Cruft will accumulate as developers fear changing working code
- Duplication is a sign of poor design / extensibility
  - Need to keep doing same kinds of things, but there’s no easy way to automate it

What you are supposed to do instead

1. Identify commonalities across code base
2. Refactor duplicate functionality to one place in the code
   - Functions with parameters
   - Base class encapsulates commonalities, derived classes specialize peculiarities
   - Generics / templates for classes / functions

Cloning is bad?

- Whoops!
- How did *that* happen?
- Have we been led astray?

Handel's Messiah

All we like sheep
All we like sheep
All we like sheep
All we like sheep
… have gone astraaaaaay

Formula, repetition, duplication

- In the arts and life, we seek to explore the new through careful venturing away from the familiar
  - Watch a toddler exploring his/her world
- The familiar can be
  - a narrative structure (e.g., a fairy tale, a knock-knock joke),
  - a chorus (new to us, but repeated),
  - a theme (e.g., sacrifice for love), …
- Humans also seem to find comfort in ritual
  - We seek refuge in the familiar when the external world seems unpredictable and frightening

Formula, repetition, duplication

- But this is *engineering*!
  - And we have no need of ritual in a utilitarian design!
- Ritual no, but repetition yes!
  - In traditional engineering, we scale up by repeated instantiations of design elements

Formula, repetition, duplication

- But this is *software* engineering!
  - We don't need duplication in a *software* design, right?


Investors Flock to VMware on First Day of Trading - New York Times http://www.nytimes.com/2007/08/15/technology/15soft.html?_r=1...

The New York Times nytimes.com

August 15, 2007

Investors Flock to VMware on First Day of Trading

By STEVE LOHR

Shares of VMware, the fast-growing Silicon Valley software maker, jumped 76 percent in their first day of trading yesterday as investors bet that the company would continue to fend off competitors like Microsoft for years to come.

VMware's initial public offering raised about $1.1 billion, making it the largest I.P.O. by a technology company since Google's in 2004. Defying a weak market, the shares closed the day at $51, or $22 higher than the offering price of $29.

The company hopes that the enthusiastic first-day reception in the stock market will also have strategic and

Formula, repetition, duplication

- Replication of trusted design elements works in software too!
  - We do need familiarity as a learning tool
  - And we can — and should! — employ duplication within a disciplined engineering process

Cloning as software design practice: Bronze an exemplar!

- The *Prototype* design pattern [GoF]
  - Used to create copies of families of complex objects
- The *Self* programming language [Ungar at al.
  - Supports evolutionary designs better than trad OO langs
  - Helps with the fragile base class problem

Cloning as software design practice

- The Rule of Three (eXtreme Programming)
  - Premature abstraction is the root of much evil!
  - Design the simplest thing that could possibly work.
- Boiler-plating is key to industrial-strength COBOL development
  - Reliable designs and working systems are golden!

Cloning is good for you! [2]

1. Forking
   - Hardware variation
   - Platform variation
   - Experimental variation
2. Templating
   - Boilerplating
   - API / library protocols
   - Generalized programming idioms
3. Customization
   - Bug workarounds
   - Replicate + specialize

1. Forking

- Often used to “springboard” new or experimental development
  - Clones will need to evolve independently
  - Big chunks are copied!
- Works well when the commonalities and difference of the end solutions are unclear.

1. Forking: Platform variation

- Motivation:
  - Different platforms ⇒ very different low level details
  - Interleaving the platform-specific code in one place may be very complex
- Advantages of cloning:
  - Each (cloned) variant is simpler to maintain
  - No risk to stability of older variants
  - Platforms are likely to evolve independently, so maintenance is likely to be “mostly independent”

1. Forking: Platform variation

- Disadvantages:
  - Evolution in two dimensions: the user requirements and the support of the platform.
  - Change to the interface level means changes to many files
- Management and long-term issues:
  - Factor out platform independent functionality as much as possible
  - Document the variation points and platform peculiarities
  - As number of platforms grows, the interface to the system effectively hardens

1. Forking: Platform variation

- Structural manifestations:
  - Cloning usually happens at the file level.
    - Clones are often stored as files (or dirs) in the same source directory
- Well known examples:
  - Linux kernel “arch” subsystem
  - Apache Portable Runtime (APR)
    - Portable impl of functionality that is typically platform dependent, such as file and network access
    - `fileio -> {netware, os2, unix, win32}`
    - Typical changes: insertions of extra error checking or API calls.
    - Cloning is clearly obvious and is documented

2. Templating

- Code embodying the desired behavior already exists
  - … but the impl. language does not provide strong support for the desired abstraction
- Linked editing or source auto-generation can be used

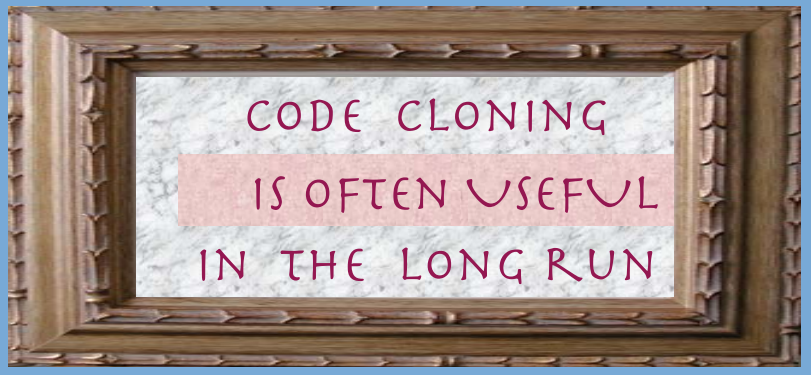
- Examples
  - COBOL boilerplate code
  - C routines that treat floats and ints analogously
  - (old) Java code that could have used generics
  - API usages for common tasks (eg GUI creation)
  - Language / platform idioms, such as safe pointer handling

3. Customization

- Existing code solves a similar problem but you can't or won't change it
  - May not own the code
  - May not want to risk change there
  - Changing may be too complex

- Examples:
  - Replicate and specialize
  - Bug workarounds

~~Myth~~ Motto



Special thanks

- To Cory Kapser, my PhD student
  - who will be graduating Real Soon Now!

References

1. http://www.cincomsmalltalk.com/userblogs/ralph/blogView?showComments=true&printTitle=Why_duplication_is_bad&entry=3271050352
2. "`Cloning considered harmful' considered harmful", Cory J. Kapser and Mike Godfrey, 2006 Working Conference on Reverse Engineering.
3. Rainer Koschke's cloning slides on the wiki are compleley awesome

All we like sheep: Cloning as an engineering tool

Michael W. Godfrey
University of Waterloo