WIKIPEDIA

Copy and paste programming

Copy-and-paste programming, sometimes referred to as just **pasting**, is the production of highly repetitive computer programming code, as produced by copy and paste operations. It is primarily a pejorative term; those who use the term are often implying a lack of programming competence. It may also be the result of technology limitations (e.g., an insufficiently expressive development environment) as subroutines or libraries would normally be used instead. However, there are occasions when copy and paste programming is considered acceptable or necessary, such as for boilerplate, loop unrolling (when not supported automatically by the compiler), or certain programming idioms, and it is supported by some source code editors in the form of snippets.

Contents

Origins

Copy and pasting is often done by inexperienced or student programmers, who find the act of writing code from scratch difficult or irritating and prefer to search for a pre-written solution or partial solution they can use as a basis for their own problem solving.[1] (See also Cargo cult programming)

Inexperienced programmers who copy code often do not fully understand the pre-written code they are taking. As such, the problem arises more from their inexperience and lack of courage in programming than from the act of copying and pasting, per se. The code often comes from disparate sources such as friends' or co-workers' code, Internet forums, code provided by the student's professors/TAs, or computer science textbooks. The result risks being a disjointed clash of styles, and may have superfluous code that tackles problems for which new solutions are no longer required.

A further problem is that bugs can easily be introduced by assumptions and design choices made in the separate sources that no longer apply when placed in a new environment.

Such code may also, in effect, be unintentionally obfuscated, as the names of variables, classes, functions and the like are typically left unchanged, even though their purpose may be completely different in the new context.[1]

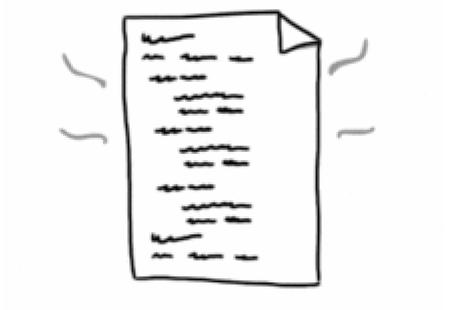
Copy and pasting in programming may also be a result of poor understanding of features common in computer languages, such as loop structures, functions and subroutines.

Duplication

Applying library code

Copy and pasting is also done by experienced programmers, who often have their own libraries of well tested, ready-to-use code snippets and generic algorithms that are easily adapted to specific tasks.[2]

Being a form of code duplication, copy-and-paste programming has some intrinsic problems; such problems are exacerbated if the code doesn't preserve any semantic link between the source text and the copies. In this case, if changes are needed, time is wasted hunting for all the duplicate locations. (This can be partially mitigated if the original code and/or the copy are properly commented; however, even then the problem remains of making the same edits multiple times. Also, because code maintenance often omits updating the comments,[3] comments describing where to find remote pieces of code are notorious for going out-of-date.)


Repetitive code being refactored by using an abstraction mechanism such as function.

Adherents of object oriented methodologies further object to the "code library" use of copy and paste. Instead of making multiple mutated copies of a generic algorithm, an object oriented approach would abstract the algorithm into a reusable encapsulated class. The class is written flexibly, with full support of inheritance and overloading, so that all calling code can be interfaced to use this generic code directly, rather than mutating the original.[4] As additional functionality is required, the library is extended (while retaining backward compatibility). This way, if the original algorithm has a bug to fix or can be improved, all software using it stands to benefit.

Branching code

Branching code is a normal part of large-team software development, allowing parallel development on both branches and hence, shorter development cycles. Classical branching has the following qualities:

- Is managed by a version control system that supports branching
- Branches are re-merged once parallel development is completed.

Copy and paste is a less formal alternative to classical branching, often used when it is foreseen that the branches will diverge more and more over time, as when a new product is being spun off from an existing product.

As a way of spinning-off a new product, copy-and-paste programming has some advantages. Because the new development initiative does not touch the code of the existing product:

- There is no need to regression test the existing product, saving on QA time associated with the new product launch, and reducing time to market.
- There is no risk of introduced bugs in the existing product, which might upset the installed user base.

The downsides are:

- If the new product does not diverge as much as anticipated from the existing product, two code bases might need to be supported (at twice the cost) where one would have done. This can lead to expensive refactoring and manual merging down the line.
- The duplicate code base doubles the time required to implement changes which may be desired across both products; this *increases* time-to-market for such changes, and may in fact wipe out any time gains achieved by branching the code in the first place.

Similar to above, the alternative to a copy-and-paste approach would be a modularized approach:

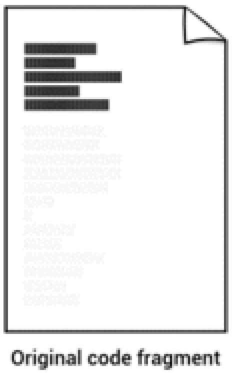
- Start by factoring out code to be shared by both products into libraries.
- Use those libraries (rather than a second copy of the code base) as the foundation for development of the new product.
- If an additional third, fourth, or fifth version of the product is envisaged down the line, this approach is far stronger, because the ready-made code libraries dramatically shorten the development life cycle for any additional products after the second.[5]

Repetitive tasks or variations of a task

One of the most harmful forms of copy-and-paste programming occurs in code that performs a repetitive task, or variations of the same basic task depending on some variable. Each instance is copied from above and pasted in again, with minor modifications. Harmful effects include:

- The copy and paste approach often leads to large methods (a bad code smell).
- Each instance creates a code duplicate, with all the problems discussed in prior sections, but with a much greater scope. Scores of duplications are common; hundreds are possible. Bug fixes, in particular, become very difficult and costly in such code.[6]

- Such code also suffers from significant readability issues, due to the difficulty of discerning exactly what differs between each repetition. This has a direct impact on the risks and costs of revising the code.
- The procedural programming model strongly discourages the copy-and-paste approach to repetitive tasks. Under a procedural model, a preferred approach to repetitive tasks is to create a function or subroutine that performs a single pass through the task; this subroutine is then called by the parent routine, either repetitively or better yet, with some form of looping structure. Such code is termed "well decomposed", and is recommended as being easier to read and more readily extensible.[7]
- The general rule of thumb applicable to this case is "don't repeat yourself".



Difficulty and risk of maintaining code written by copy-paste programming

Deliberate design choice

Copy and paste programming is occasionally accepted as a valid programming technique. This is most commonly seen in boilerplate, such as class declarations or importing standard libraries, or in using an existing code template (with empty contents or stub functions) as a framework to fill in.

Use of programming idioms and design patterns are similar to copy and paste programming, as they also use formulaic code. In some cases this can be expressed as a snippet, which can then be pasted in when such code is necessary, though it is often simply recalled from the programmer's mind. In other cases idioms cannot be reduced to a code template. In most cases, however, even if an idiom can be reduced to code, it will be either long enough that it is abstracted into a function, or short enough that it can be keyed in directly.

The Subtext programming language is a research project aimed at "decriminalizing" cut and paste. Using this language, cut and paste is the primary interaction model, and hence not considered an anti-pattern.

Example

A simple example is for a loop, which might be expressed as `for (int i=0; i!=n; ++i) {}`.

Sample code using such a for-loop might be:

```
void foo(int n) {
    for (int i=0; i!=n; ++i) {
       /* body */
    }
}
```

The looping code could then have been generated by the following snippet (specifying types and variable names):

```
    for ($type $loop_var = 0; $loop_var != $stop; ++$loop_var) {
        /* body */
    }
```

See also

- Cargo cult programming

References

1. "Revisiting Novice Programmers Errors" (http://portal.acm.org/citation.cfm?id=1272896&dl=GUIDE&coll=GUIDE&CFID=30889213&CFTOKEN=55540185). acm.org. Retrieved 2008-06-04.
2. "Building ASP.NET Web Pages Dynamically in the Code-Behind" (http://www.codeproject.com/KB/aspnet/dynamic_page_building.aspx?display=Print). codeproject.com. Retrieved 2008-06-04.
3. Spinellis, Diomidis. "The Bad Code Spotter's Guide" (http://www.informit.com/articles/article.aspx?p=457502). InformIT.com. Retrieved 2008-06-06.

4. Lewallen, Raymond. "4 major principles of Object-Oriented Programming" (http://codebetter.com/blogs/raymond.lewallen/pages/59908.aspx). codebetter.com. Retrieved 2008-06-04.
5. Eriksen, Lisa. "Code Reuse In Object Oriented Software Development" (http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2004/Eriksen2004.pdf) (PDF). Norwegian University of Science and Technology, Department of Computer and Information Science. Retrieved 2008-05-29.
6. Ashley Marsh. "Coding Standards – The Way to Maintainable Code" (https://www.maansoftwares.com/blog/development/coding-standards-—-way-maintainable-code). MAAN Softwares INC. Retrieved 2018-04-10.
7. "Stanford University, CS 106X ("Programming Abstractions") Course Handout: "Decomposition" " (https://web.archive.org/web/20080516005005/http://www.stanford.edu/class/cs106x/handouts/14-Decomposition.pdf) (PDF). Stanford University. Archived from the original (http://www.stanford.edu/class/cs106x/handouts/14-Decomposition.pdf) (PDF) on May 16, 2008. Retrieved 2008-06-04.

External links

- c2:CopyAndPasteProgramming
- Andrey Karpov. Consequences of using the Copy-Paste method in C++ programming and how to deal with it (http://software.intel.com/en-us/articles/consequences-of-using-the-copy-paste-method-in-c-programming-and-how-to-deal-with-it)
- Andrey Karpov. The Last Line Effect (https://medium.com/@Code_Analysis/7b1cb7f0d2a1)
- PMD's Copy/Paste Detector (http://pmd.sourceforge.net/pmd-4.2.5/cpd.html), CPD.