Using Artificial Intelligence (AI)

For Large Software Engineering Projects – Part 4

Capers Jones

Foreword: The following pages constitute Part 4 of *Using Artificial Intelligence for Large Software Engineering Projects* by Capers Jones, released for distribution to the International Function Point Users Group (IFPUG) in January of 2025. Topics in this extract include: Reuseable Components and AI.

Acknowledgment: The IFPUG community expresses its appreciation to Mr. Jones for his lifelong pursuit of metrics-based state of and recommended improvements for software development practices globally.

Joe Schofield

Software Development from Certified Reusable Components and Artificial Intelligence

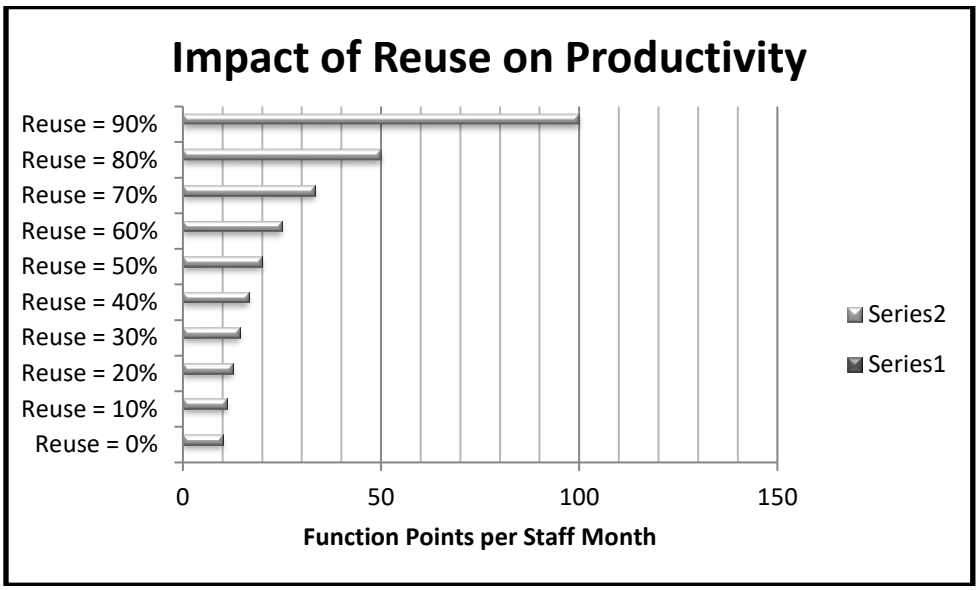
Because custom designs and manual coding are slow, error prone, and inefficient it is useful to show the future impacts of varying levels of reusable components. The phrase "reusable components" refers to much more than just reusable source code. A basic feature of AI for software is a full library of reusable materials. Samples what needs to be reused are shown in table n:

Table 4: Major Reusable Software Components

1. Reusable requirements
2. Reusable architecture
3. Reusable design
4. Reusable project plans
5. Reusable estimates
6. Reusable source code
7. Reusable test plans
8. Reusable test scripts
9. Reusable test cases
10. Reusable marketing plans
11. Reusable user manuals
12. Reusable training materials
13. Reusable HELP screens and help text
14. Reusable customer support plans
15. Reusable maintenance plans

Figure 1 illustrates why software reuse is the ultimate future software engineering methodology that is needed to achieve high levels of productivity, quality, and schedule adherence at the same time. Figure 1 illustrates a generic application of 1,000 function points coded in the Java language:

Figure 1: Impact of Reuse on Software Productivity



Applications of 1000 function points in size are normally created at rates of between about 6.00 and 13.50 function points per staff month using custom designs and manual coding. Waterfall would be at the low end of the spectrum while agile, RUP, TSP, and other advanced methods would be at the high end of the spectrum.

Here are a few samples of various development methods and associated productivity rates using function points per staff month for applications of a nominal 1000 function points in size assuming above-average team experience levels:

Table 4: Productivity Ranges with and without Software Reuse

	Methodologies	**Function Points Per Staff Month**
1.	AI generated with 90% reuse	300.00
2.	Mashup with 65% reuse	47.41
3.	Hybrid with 50% reuse	19.71
4.	Hybrid with 25% reuse	15.52
5.	Agile/scrum	12.08
6.	Spiral at CMMI® Level 5	12.05
7.	Extreme Programming (XP)	11.89
8.	TSP at CMMI® Level 5	11.54
9.	Rational Unified Process (RUP)	9.92
10.	Iterative at CMMI® Level 3	9.37
11.	Iterative with object-oriented methods	9.31

12. Lean six-sigma	9.21
13. Waterfall domestic outsource	6.80
14. Waterfall offshore outsource	6.29
15. Waterfall at CMMI® Level 1	6.05
16. Waterfall with novices at CMMI® 1	5.03

As can be seen the examples with reuse are at the top of the list. Below the top two examples zero reuse is assumed. It should be noted than none of the reuse shown above was "certified" as discussed in this article. Primarily the reusable components came from similar applications within the same company. No doubt some of these reusable materials contained bugs, security flaws, or both. Uncertified reuse is cheaper than custom development, but also somewhat hazardous.

However, without reuse no method would top about 15.00 function points per staff month for applications of a nominal 1000 function points in size. This is much slower than needed for rapidly changing business situations. It is somewhat analogous to having a national automobile speed limit of only 25 miles per hour.

Figure 1 illustrates reuse from 0% to 90% which is likely to be the upper limit for many applications. If it were possible to create new applications from 100% reusable components productivity could top 150 function points per staff month, or about 15 times faster than today's averages in 2024 even for agile projects.

Reuse also benefits quality and security. Table 5 shows the approximate impact of reuse on delivered software defects for an application of 1000 function points in size. Defect potentials are shown in terms of defects per function point because that metric allows all defect origins to be included (requirements defects, design defects, code defects, document defects, and bad fixes or secondary defects):

Table 5: Software Reuse and Software Quality Levels at Delivery

Percent of total Reuse	**Defect Potential per Function Pt.**	**Defect Removal Percent**	**Delivered Defects per FP**
90.00%	1.00	99.50%	0.01
80.00%	1.25	98.00%	0.03
70.00%	1.50	97.00%	0.05
60.00%	2.00	95.00%	0.10
50.00%	2.50	92.00%	0.20
40.00%	3.00	90.00%	0.30
30.00%	3.75	87.00%	0.49

20.00%	4.25	85.00%	0.64
10.00%	5.00	83.00%	0.85
0.00%	5.50	80.00%	1.10

As clearly shown by table 5 software reuse will have major benefits for software quality improvement.

Table 6 shows the same sequence as Table 5 only for the prevention and removal of security flaws, also for an application of 1000 function points in size. In general, there are fewer security flaws than defects, but they are harder to find and to eliminate so the defect removal efficiency is lower against security flaws than against ordinary bugs:

Table 6: Reuse and Software Security Flaws at Delivery

Percent of Reuse	**Security Flaws per Function Pt.**	**Flaw Removal Percent**	**Delivered Flaws per FP**
90%	0.40	99.00%	0.004
80%	0.50	96.00%	0.020
70%	0.60	92.00%	0.048
60%	0.80	90.00%	0.080
50%	1.00	87.00%	0.130
40%	1.20	83.60%	0.197
30%	1.50	80.75%	0.289
20%	1.91	78.85%	0.404
10%	2.25	76.95%	0.519
0%	2.85	75.05%	0.711

The bottom line is that certified reusable components would be substantially free from both latent defects and also from latent security flaws.

Reuse potential volumes vary by industry and application type. Reuse potential is the percentage of overall application features that are provided by certified reusable components rather than being custom designed and manually coded. Table 7 shows approximate reuse potentials for the current year of 2024, and then future reuse potentials for 2054 or thirty years from now:

Table 7: Software Reuse Potentials by 2054

		2024 Reuse Potential	2054 Reuse Potential
1	AI-generated software systems Telecommunications applications	30.00%	97.00%
2	Embedded applications (automotive)	20.00%	97.00%
3	Electric power applications	35.00%	95.00%
4	Airline applications (reservations, logistics)	15.00%	95.00%
5	Hotel applications (reservations, logistics	18.00%	95.00%
6	Medical applications billing	15.00%	95.00%
7	Insurance applications - property	45.00%	90.00%
8	Insurance applications - life	50.00%	90.00%
9	Banking applications	60.00%	85.00%
10	State government applications	35.00%	85.00%
11	Education applications - primary/secondary	30.00%	85.00%
12	Wholesale applications	60.00%	85.00%
13	Municipal government applications	40.00%	80.00%
14	Retail applications	40.00%	80.00%
15	Manufacturing applications	45.00%	75.00%
16	Federal civilian government applications	30.00%	75.00%
17	Weapons systems	20.00%	75.00%
18	Insurance applications - health	25.00%	70.00%
19	Education applications - university	35.00%	70.00%
20	Medical applications - diagnostic	5.00%	55.00%
	Average Reuse Potential	**32.65%**	**83.70%**

For many industries most corporate software applications do pretty much the same thing as every other company in the industry. The concept of reusable components is to identify the specific sets of features that are potentially reusable for every company in specific industries. For some industries such as banking and stock trading, there are Federal laws and mandates that make reuse mandatory for at least some critical features.

Some examples of common reusable features circa 2023 include but are not limited to the following: 1) accounting rate of return, 2) automotive GPS software, 3) bar code reading devices, 4) browser add-ins, 5) compound interest, 6) PBX switches; 7) Crystal reports, 8) cryptographic key processing, 9) currency conversion, 10) Excel functions,11) facial recognition, 12) inflation rates, 13) internal rate of return, 14) metrics conversion, 15) PDF document conversion, 16) real estate depreciation, 17) state sales tax calculations, 18) traffic light controls, and 19) Word templates; 20) World-time clock features.

As of today, reusable components approximate 15% of the features in many common applications, and sometimes top 30%. As of 2024 reuse is not always certified, but the major commercial reusable components are fairly reliable. Unfortunately there are several gaps in the reuse domain that need to be filled: 1) There is no effective taxonomy of reusable features; 2) There are no available catalogs of reusable features that might be acquired from commercial sources; 3) Software measurements tend to ignore or omit reusable features, which distorts productivity and quality data; 4) Some software estimating tools do not include reuse (although this is a standard feature in the author's Software Risk Master (SRM) estimating tool; 5) Much of the literature on reuse only covers code and does not yet fully support reusable requirements, reusable designs, reusable test materials, and reusable user documents.

One major barrier to expanding reuse at the level of specific functions is the fact that there are no effective taxonomies for individual features used in software applications. Current taxonomies work on entire software applications, but are not yet applied to the specific feature sets of these applications. For example, the widely used Excel spreadsheet application has dozens of built-in reusable functions, but there is no good taxonomy for identifying what all of these functions do.

Obviously, the commercial software industry and the open-source software industry are providing reuse merely by selling software applications that are used by millions of people. For example, Microsoft Windows is probably the single most widely used application on the planet with more than a billion users in over 200 countries. The commercial and open-source software markets provide an existence proof that software reuse is an economically viable business.

Commercial reuse is fairly large and growing industry circa 2024. For example, hundreds of applications use Crystal Reports. Thousands use commercial and reusable static analysis tools, firewalls, anti-virus packages, and the like. Hundreds of major companies deploy Enterprise Resource Planning (ERP) tools which attempt reuse at the corporate portfolio level. Reuse is not a new technology, but neither is it yet an industry with proper certification to eliminate bugs and security flaws prior to deployment.