Using Artificial Intelligence (AI) For Large Software Engineering Projects – Part 5

Capers Jones

Foreword: The following pages constitute Part 5 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: Predicted Tools and Techniques for the Next 30 Years; and Development Suites, Development Stages, AI in Software; Conclusions, Reports and Books on Artificial Intelligence.

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

What Software Engineering Project Managers and Developers Will Have by 2054

If you consider the common problems of major software projects such as outright cancellation, schedule slips, and cost overruns it is clear that project managers and developers need better tools and better data than is common today.

What will probably be available within 10 years are these fully integrated capabilities that combine predictive analytics, a full suite of project measures and advanced development from a combination certified reusable materials and artificial intelligence tools:

1. Early sizing and estimating prior to full requirements via pattern matching (available today).
2. Formal taxonomies of both applications and feature sets.
3. Automated metrics conversion between function point methods, story points, etc. (available today).
4. Early predictions of probable requirements creep (available today).
5. Early predictions of defects in requirements, design, code, and other sources (available today).
6. Early predictions of probable security problems in software (available today).
7. Early predictions of defect removal efficiency (DRE) against all defect types (available today).
8. Early production of animated graphical project plans integrating requirements creep.
9. Early acquisition and analysis of benchmarks of similar projects already completed.
10. Early analysis of all similar projects completed within the past 5 years.
11. Early acquisition of data on security attacks against similar projects already completed.
12. Early security plan to avoid security flaws and optimize data security.
13. Early predictions of necessary test cases and test scripts (available today).
14. Early predictions of all necessary documents, sizes, and costs (available today).
15. Early predictions of full staffing needs including specialists such as project office staff (available today).
16. Early risk analysis of all potential application risks prior to full requirements (available today).
17. Early analysis of both available reusable components and novel components needing custom development.
18. Dynamic, interactive, animated architecture and design documents for performance and security analysis.
19. Acquisition of certified reusable components from commercial sources.
20. Nomination of selected new features to become new certified reusable components.
21. Semi-automatic construction of applications from certified reusable components.
22. Semi-automatic construction of test scripts and test plans.
23. Semi-automatic testing from unit test through system test.
24. Semi-automatic tracking of project effort and costs as they accumulate (available today).
25. Semi-automatic logging and tracking of defects found via inspection, static analysis, and testing (available today).
26. Daily refreshed cost-to-complete and time-to-complete estimates.
27. Daily refreshed quality and defect removal efficiency (DRE) estimates.
28. Running FOG and Flesch indices on all text documents to ensure clarity and readability.
29. Formal inspections of all critical requirements and design features.
30. 3D design images in full color with animation.
31. 3D printers for generating actual objects designed with 3D tools.
32. Running text static analysis on all text documents to find errors and contradictions.
33. Running code static analysis on all code changes on a daily basis.
34. Daily runs of cyclomatic and essential complexity of all code segments.
35. Daily runs of test coverage tools when testing begins.
36. Daily status reports to clients and executives that highlight any critical issues.
37. Achieving full licensing and board certification for software engineering specialists.
38. Achieving defect potentials < 2.00 per function point.

39. Achieving test coverage of > 95.0% for risks and paths.
40. Achieving defect removal efficiency (DRE) > 99.5%.
41. Achieving security flaw removal efficiency (SRE) > 99.9%.
42. Achieving productivity rates > 100 function points per month.
43. Achieving development schedules < 24 months for 10,000 function points.
44. Achieving cancellation rates of < 1.0% for 10,000 function points.
45. Achieving maintenance assignment scopes > 5,000 function points.
46. Achieving certified reuse volumes > 85%.
47. Achieving mean-time-to-failure of > 365 days.
48. Achieving schedule and cost overruns of < 1.0% of plan for all applications.
49. Achieving blockage of cyber-attacks of > 99.99%.
50. AI combined with full reuse may raise productivity of > 1000 FP per month.
51. AI combined with zero-defect reusable components my lower defect potentials.
52. AI combined with full reuse may lower the schedules for large systems from over three years to less than 3 days.

Many of these software engineering needs are already available today. even if not yet widely deployed. The toughest needs are those associated with fully certifying the reusable materials and making sets of reusable materials available as needed.

The cyber security goals are also tough, in part because cyber security during development is still not a critical-path topic for many companies and government agencies. The impact or artificial intelligence may be larger than all of the others but it is too early in 2024 to have empirical data on AI software development.

Paul Starr's excellent book The Social Transformation of American Medicine (Pulitzer Prize in 1982) shows the path followed by the American Medical Association (AMA) to improve medical education and weed out malpractice. n This book should be on the shelf of every software engineer and software engineering academic because we are still far behind medicine in terms of professionalism, licensing, board certification for recognized specialties. As of 2023 the Department of Commerce still ranks software as a craft and not as an engineering field.

Custom designs and manual coding are intrinsically slow, expensive, and error prone and nothing will change that fact. Software construction from certified reusable components is the only known method of elevating productivity, quality, and security at the same time.

Development of Suites for Reusable Materials

Although reusable materials have major benefits when deployed, the construction of reusable materials and their certification normally is more expensive than ordinary custom development due to the need for rigorous quality control and security flaw prevention (and removal). In general, developing a reusable component takes about 50% longer and is about 75% more expensive than developing the same feature using normal development practices. The sequence for developing standard sets of reusable components includes but is not limited to the following:

Development Stages for Certified Reusable Components

1. Sizing of each feature using function points, SNAP, and logical code size
2. Planning and estimating schedules and costs of feature construction
3. Planning for potential risks and security attacks for each feature
4. Market analysis of reuse potential for each feature (from 10 to 1,000,000 uses)
5. Make or buy analysis for certified reusable materials
6. Tax and government restriction analysis for all reusable materials
7. Mining legacy applications for potential reusable components
8. Patent analysis for unique or valuable intellectual property
9. Formal and reusable requirements creation for each feature
10. Formal and reusable design creation for each feature
11. Formal and reusable code creation for each feature
12. Formal and reusable test plan creation for each feature
13. Formal and reusable test script creation for each feature
14. Formal and reusable test case creation for each feature
15. Formal and reusable user training material for each feature
16. Formal and reusable user documentation for each feature
17. Formal translation into foreign languages, if needed for global sales
18. Formal and reusable HELP text for each feature
19. Formal security inspection for each feature
20. Formal usability inspection for each feature
21. Formal quality inspection for each feature
22. Formal text static analysis
23. Formal running of FOG or FLESCH readability tools on all text
24. Formal code static analysis for all source code
25. Formal mathematical test case construction for each feature's test cases
26. Formal measurement of cyclomatic complexity for all code
27. Formal measurement of test coverage for reusable test cases
28. Formal testing of each reusable code segment for errors and security flaws
29. Formal collection of historical data (quality, costs, etc.) for each feature
30. Formal certification of each feature prior to integration in component library

As can be seen almost half of the development stages for constructing reusable materials would probably not be performed for ordinary components that are not specifically aimed at being entered into libraries of reusable materials. This is why development of certified reusable components is slower and more expensive than ordinary development. The high initial costs are of course offset by the value and much lower costs of each subsequent reuse of the component.

Because the costs and schedules of creating and certifying reusable software components are much larger than for ordinary development, there is no economic value from creating these components unless they are likely to be reused many times. This brings up a point that in the United States some reusable materials may be taxable assets by the Internal Revenue Service (IRS). Companies should check with tax accountants and tax attorneys prior to beginning a formal reuse program.

It is likely that some kind of brokerage business might be needed to handle the aggregation and distribution of reusable materials. Specific companies are probably not set up or qualified to market their own reusable materials.

A reuse clearing house might be created that could purchase reusable software materials at wholesale prices and then remarket the materials at retail prices. For example, reusable banking modules might be purchased at a cost of $1,500 per function point from individual banks, and then sold to other banks for $150 per function point.

Since it would probably cost each bank more than $1000 per function point for custom development, being able to acquire key banking features for only $150 per function point would be a significant cost saving. Of course, the reuse clearing house would need to sell more than a dozen copies of each reusable component in order to make a profit. Since banking is a major industry, each reusable component might be sold to hundreds or even thousands of banks.

Developing Artificial Intelligence Software Construction Tools

It is too soon in 2024 to judge the effectiveness of artificial intelligence for software development, but in other fields where AI is being applied the AI results are between 10 and 100 times faster than the conventional methods they replaced.

It is likely that AI applied to software engineering will have similar benefits.   It is unknown in 2024 whether the quality problems of large software applications can be solved by artificial intelligence.

Another interesting thought about artificial intelligence is when AI is used for software engineering it can also generate user manuals and training materials at the same time.

Summary and Conclusions on the Future of Software Engineering

For historical reasons software development methods assume that all applications are unique. This is not true today, although it was true in the 1960's and 1970's. The majority of software applications today within specific industries such as banking, insurance, stocks, energy, etc. are as much alike as peas in a pod. These similarities could lead to much faster development with better quality and higher security levels if certified reusable components began to move into the main stream. Custom designs and manual coding are intrinsically slow, expensive, error prone, and vulnerable to security flaws no matter what development methods are used and what programming languages are utilized for coding. So long as software engineering depends upon custom designs and manual programming it will never be a true engineering profession and software quality and reliability will be hazardous, as they are today.

A synergistic combination of a formal taxonomy of software features, dynamic 3D visual design methods, and ever-growing libraries of certified reusable components embedded with 3D software development tools have the theoretical potential for increasing software development productivity rates from today's norms of less than 10 function points per staff month to more than 350 function points per staff month by 2054.

When artificial intelligence (AI) combined with reuse are considered, then by 2054 large systems in the 10,000-function point size range may be created in 3 weeks instead of 3 years and achieve production rates in excess of 1,000 function points per staff month. Two of the three weeks would deal with user requirements; development would take 3 days; and the final 2 days would be devoted to trying out the new application before it enters production.

An unanswered question about AI and software is whether AI can also improve the quality of large systems. Yet another unanswered question is whether or not AI developed applications will have the same rate of changing requirements after delivery. Probably requirements changes will be as large as ever because they are due to changes in external business and technical issues and not in the software itself.

Today the labor content of software is much higher than it should be. Quality and reliability are much worse than they should be. Security flaws and cyber-attacks represent one of the major business threats in all history and improvements are urgently needed. Certified reuse has the potential for improving speed, costs, quality, and security at the same time.

It is technically possible that everything in this paper could be done in 2044 instead of 2054, but probably another 20 years will pass before a synergistic combination of formal taxonomies, pattern matching, dynamic animated software planning tools, animated design engines, certified reusable components, intelligent agents based on artificial intelligence, and formal risk analysis.

This final image shows a programming office in 2055 with humans and robots all working on a large and complex system of around 100,000 function points:

Artificial intelligence can improve productivity by over 1000% and reduce defects in delivered software by over 99% compared to 2024.

RECENT REPORTS AND ARTICLES ON ARTIFICIAL INTELLIGENCE

Five Ways Artificial Intelligence Will Change the World	NBC News 2023
The Future of AI's Impact on Society	MIT Technology Review 2023
How Will AI Impact the Future of Work	Forbes 2023
Artificial Intelligence News	Science Daily 2023
Artificial Intelligence (AI) Technology	The Guardian 2023
Artificial Intelligence	BBC News 2023

BOOKS ABOUT ARTIFICIAL INTELLIGENCE

The McGraw Hill Illustrated Encyclopedia of Robotics and Artificial Intelligence, McGraw Hill 2022.

Fundamental of Artificial Intelligence: Problem Solving and Automated Reasoning; Miroslav Kubert, McGraw Hill, 2023

The Essence of Artificial Intelligence; Alison Crowly, Prentice Hall; 2023

Philosophy & Artificial Intelligence; Todd C. Moody; Prentice Hall 2023

Artificial Intelligence: A Modern Approach; Stuart Russel and Peter Norvig; Pearson; 2022