Software Engineering Process Revolution

Chi-Hung Kao\textsuperscript{1}, Jay Xiong\textsuperscript{2}

\textsuperscript{1} NSEsoftware., LLC, USA
\textsuperscript{2} Northwestern Polytechnic University

Abstract - This article introduces the NSE (Nonlinear Software Engineering paradigm) process model based on complexity science indicating that almost all tasks/activities are performed holistically and globally. Some applications show that the techniques and supporting platforms of NSE process model can not only make revolutionary changes to almost all aspects in software engineering for efficiently handling software complexity, invisibility, changeability, and conformity, but solve the critical problems such as low productivity and quality, high cost and risk, existing with the old-established software engineering paradigm. The NSE helps software developers raising productivity, dropping costs, and removing dramatic amount of the defects in their products.

Keywords: software process model, software engineering revolution, methodology, testing, quality assurance, productivity, maintenance

1. Almost All of the Existing Software Engineering Process Models Are Outdated

Almost all existing software engineering process models, no matter if they are waterfall models, incremental development models, iterative development models, or a new one recommended by Alistair Cockburn combining both incremental and iterative development together\cite{1}, are outdated because they are linear models with only one track forward in one direction without upstream movement at all, requiring software developers and the customers always do all things right without making any mistake or any wrong decision – but it is impossible, complying with the superposition principle that the whole of a system is the sum of its parts, so that almost all tasks/activities are performed linearly, locally and partially, making the defects introduced into a software product at the upper phases easy to propagate to the lower phases and the defect removal cost increase tenfold several times.

The common drawbacks of the existing software process models also include:

(a) None of them are created to efficiently handle the essential issues existing with software products – complexity, invisibility, changeability, and conformity, defined by Brooks\cite{2}.

(b) None of them are able to efficiently solve the most critical problems with software development - low quality and productivity, and high cost and risk.

(c) None of them are able to make significant improvement to the software project success rate, so that today the software project success rate is still at about 30\%\cite{3} - it is unacceptable in any other industry.

(d) Incomplete - None of them are able to efficiently support software maintenance which takes 75\% or more of the total effort and cost for software product development\cite{4}, because they do not satisfy the following listed essential conditions for an efficient software maintenance support:

(1) being able to greatly reduce the amount of defects introduced into a software product and the defects propagated to the software maintenance phase through defect prevention and defect propagation prevention;

(2) being able to help users perform software maintenance holistically and globally;

(3) being able to help users prevent the side-effects for the implementation of requirement changes or code modifications;

(4) being able to provide necessary means to help users greatly reduce the time, cost, and resources in regression testing after software modification, such as the capability for test case minimization, and intelligent test case selection through backward traceability from a modified module or segment (a set of statements with the same execution conditions);

(5) being able to help the customer side maintain a software product developed by others with almost the same conditions as it is maintained by the product development side (see table 2 about the “Software” definition).
NSE process model with “two-way iteration and multiple tracks” satisfies the five conditions. It is possible for the NSE process model with the support techniques and tools to help software development organizations reduce 2/3 of the total effort and total cost spent in software maintenance - equal to double their productivity and halve their cost. It is important to point out that with NSE there is no major difference between the software development process and the software maintenance process, because:

* both processes support requirement changes and code modifications with side-effect prevention through various bidirectional traceabilities.

* When the NSE nonlinear process model is followed, the quality of a software product is ensured from the first step (see section 7) down to the last step in maintenance through defect prevention and defect propagation prevention, so that the defects propagated to the maintenance phase are greatly reduced.

2. The foundation of NSE and the NSE process model – complexity science

Complexity science has been called the science of the 21st century by Stephen Hawking and Edward O. Wilson.

The essential principles of complexity science complied with by the NSE process model include the:

Nonlinearity principle
Holism principle - that all the properties of a given system cannot be determined or explained by its components alone. Instead, the characteristics and behavior of the whole of a complex system emerge from the interaction of its components and the interaction between it and the environment.

Initial Condition Sensitivity principle
Sensitivity to Change principle
Dynamics principle
Openness principle
Self-organization principle, and
Self-adaptation principle

NSE engineering process model is innovated through the use of a paradigm-shift framework, FDS (the Five-Dimensional Structure Synthesis method - a paradigm-shift framework innovated by Jay Xiong) as shown in Fig. 1.

3. Advanced techniques innovated to support NSE and the NSE process model

Fourteen advanced software engineering techniques are innovated to support NSE process model as shown in Fig. 2. Our two related papers titled as “Automated and Self-maintainable Traceability” and “Software Testing Revolution” are accepted by CrossTalk for publication. About the other 12 techniques, please see table 2, or read our published article titled as “A complete revolution in software engineering based on complexity science”[5]

4. Description of the NSE process model
Number section and subsection headings consecutively in numbers and type them in bold. Use point size 14 for section headings and 12 for subsection headings and 10 for subsection within a subsection.

The NSE process model (Fig. 3) consists of the pre-process part and the main process part which is supported by a facility for automated and bi-directional traceability using Time Tags for data mapping and bookmarks for opening a document traced from the corresponding location.

**The NSE process model** (Fig. 3) consists of the pre-process part and the main process part which is supported by a facility for automated and bi-directional traceability using Time Tags for data mapping and bookmarks for opening a document traced from the corresponding location.

**The NSE Process Model and the automated and self-maintainable traceability facility**

The objectives of the pre-process are:

- **a)** Assigning priority to the requirements for better control of the development schedule and the budget, implementing the important requirements earlier;

- **b)** Performing prototyping design and evaluation for some unfamiliar requirements to reduce project development risk;

- **c)** Performing function decomposition of the functional requirements using the Holistic, Actor-Action and Event-Response driven, Traceable, Visual, and Executable technique (HAETVE) to replace the Use Case approach which is not holistic, and the results obtained are not traceable and not directly executable for defect removal.

- **d)** Making a primary version of the requirement specification document using standard templates provided to prevent defects of missing something;

- **e)** Carrying out Synthesis Design of the system using the “Dummy programming” technique through the use of dummy modules to complete a dummy system. According to the Generative Holism Theory of complexity science, the whole of a complex system may not be “built” from its components, but exists (like a human embryo) earlier than its parts, then “grows up” with its parts;

- **f)** Organizing the document hierarchy using bookmarks, including the test scripts and the test case numbers, so that when there is a need to modify a requirement, it is easy to find the related test scripts and the test cases to perform forward tracing to find the related documents and the source code.

**The Major Steps Of The Main Process**

**Step 1**: According to the project development plan, the priority assigned to the requirements, take one or a set of requirements to implement visually. It is recommended to select the critical and essential requirements (about 20% of the initial requirements) first to implement and form an essential version of the software product through incremental integration development to make a software system grow up incrementally.

The NSE process model supports the NSE software development methodology based on Generative Holism and driven by defect prevention and various traceabilities - see Fig. 4.
Fig. 4 NSE software development methodology

Step 2: Apply the Synthesis Design and Incremental Implementation, Iteration, and Integration Technique with the Holistic And Traceable Diagram Generation technique to further perform preliminary design for the selected requirement(s) according to the detailed requirement specification to improve the corresponding part of the dummy system obtained in the preprocess phase, then perform formal inspection and review using traceable documents, and design the corresponding test cases to dynamically test the result of the preliminary design using the Transparent-box method (see table 2) to prevent inconsistency defects through bidirectional traceability established automatically. After that, perform detailed design for the selected requirement(s) according to the result of the preliminary design with formal inspection and review using traceable documents, and dynamic testing like what was done in the preliminary design process. If something critical is found, go to the requirement development phases, or if the solution method does not satisfy the requirement(s), go back to the preprocess.

Step 3: Apply the Synthesis Design and Incremental Implementation, Iteration, and Integration Technique to perform incremental coding: on the generated system decomposition chart, highlight the corresponding key module(s) and the related modules for the selected requirement(s), then assign an incremental bottom-up coding order to the modules automatically with the NSE support platform. When we are writing a function call statement to a called module which has been coded according to the order assigned, we can read the diagrammed source code in another window to know how many parameters are needed, their types, and their sequence to prevent inconsistency defects between the module interfaces.

If something critical is found in the coding process, go to the upper phases through backward tracing, or if the solution method does not satisfy the requirement(s), go back to the preprocess again.

Step 4: Perform incremental unit testing with integration testing, and finally system testing, mainly using the transparent-box method to combine functional and structural testing together. At the same time, perform MC/DC (Modified Condition/Decision Coverage) test coverage analysis, performance analysis, memory leak analysis and memory usage violation check. According to the incremental coding and unit testing order, when we code a module, all modules called by it must have been coded already so that there is no need to design and use a stub module to replace a called module – in this way the unit testing also becomes integration testing with all modules being called together.

If something critical is found in the testing process, go to the upper phases through backward tracing, or if the solution method does not satisfy the requirement(s), go back to the preprocess again.

In the system testing process, NSE offers the capability to capture users’ GUI operations and play them back automatically for regression testing, and the capability for MC/DC test coverage analysis for the entire product, plus performance analysis, test case efficiency analysis and test case minimization for efficient regression testing after code modification. With system testing, an automated and bidirectional traceability among all artifacts including the source code will be established for defect prevention.

Step 5: Perform systematic, disciplined, and quantifiable software maintenance using the Holistic, Global, and Side-Effect-Prevention Based Software Maintenance Technique:

1. Respond to requirement changes and new requirements or code modifications in real-time to implement them holistically and globally with side-effect prevention.

2. Bring great savings to regression testing after requirement changes or code modification through test case efficiency analysis and test case minimization, plus intelligent test case selection through backward traceability between test cases and the source code.

3. Make it possible to reduce the cost and effort spent in software maintenance from 75% or more of the total with the old-established paradigm to about 25% of the total with NSE through side-effect prevention. If there is still something wrong after the implementation of requirement change or code modification, perform intelligent version comparison to help users locate the defects in system-level, file-level, module-level, and statement level.
Step 6: Closely combine the project management process and the product development process together, making the project plan, schedule charts, and cost estimation reports traceable with the requirement implementation and the source code, for better control of the cost and project development schedule.

Step 7: Establish a project web site and the related technical forum for real time communication and technical discussion among team members to report progress of the project, and to open technical discussions for brainstorming, reporting a variety of related events, error handling processes and results, and especially unexpected events in order to discuss the response, which can all be traced back through the bidirectional and automatic traceability mechanism to update them in real time.

Step 8: Frequently deliver working products to the customer for review and evaluation, even if there is no real output for a dummy system designed in the requirement development phase. Get the customer’s feedback to improve the product development.

5. Quality assurance with NSE process model – NSE SQA

The quality assurance priority of NSE-SQA is as follows:

a) Defect prevention in all phases for preventing repeatable defects and possible new defects

b) Defect propagation prevention (removing defects from the source) through:

(1) semi-automated inspection and walkthrough using traceable artifacts and diagrammed source code; (2) transparent-box testing in all phases

c) Refactoring based on complexity analysis (20% of complex code causes 80% of the defects)

d) Deep and broad testing, and quality measurement.

5.1. Highlights of the NSE-SQA

(a) based on defect prevention, defect propagation prevention, inspection and review in the entire lifecycle using traceable documents and source code, refactoring based on complexity measurement and performance analysis, and deeper and broader software testing plus quality measurement

(b) possible to remove 99% to 99.99% of defects of a software product - a detailed comparison on defect removal efficiency is shown in table 1 (The data reported by SPR [6] through the analysis of more than 12,000 projects with the old-established software engineering paradigm is shown in italic; data with NSE SQA is shown in bold).

6. The major features of NSE process model

The major features and characteristics of the NSE process model include:

(1) Dual-process: NSE model consists of the preprocess and the main process. They are different but also closely linked together.

(2) Nonlinear: The NSE model is nonlinear, complying with the Nonlinearity principle and the Holism principle.

(3) Parallel with Multiple tracks: “Much of software architecture, implementation, and realization can proceed in parallel.” [7]. For reducing waiting time and speeding up software development processes, the NSE process model supports tasks being performed in parallel with multiple tracks through bidirectional traceability.

(4) Real time: “Timely updating is of critical importance.” [8].

(5) Incremental development with two-way iteration: The NSE process model supports incremental development with two-way iteration, including refactoring to handle highly complex modules and performance bottlenecks with side-effect prevention. When a critical issue is found in the main process, the work flow may go back to the preprocess for selecting a better solution method, and so on.

(6) The software development process and software maintenance process are combined together seamlessly: With the NSE process model, there is no big difference between the software development process and the software maintenance process – both support requirement changes through side-effect prevention.

(7) The software development process and the project management process are combined together closely: all documents including the project management documents such as the project development plan, the schedule chart, and the cost estimation report are traceable with the requirement implementation and the source code for better
control of the product development. NSE process model also supports the critical requirements and most important requirements being implemented early with the assigned priority to avoid budget overuse – if necessary, some optional requirements and not so important requirements can be ignored temporarily.

(8) Adaptation focused rather than predictability focused: the entire world is always changing, so the NSE process model is adaptation focused rather than predictability focused – it supports requirement changes, code modifications, data modifications, and document modifications to make them consistent and updated with side-effect prevention.

(9) Defect prevention driven

(10) People are considered as the first order driver for software development - When people consider “people as the first-order” to software development, they focus on how to trust and support people better for their jobs, but ignore the other side of people’s effect on software development – almost all defects introduced into software products are made by people, the developers and the customers. So NSE supports people in two ways: one is to support them with better methodology, technology, and tools; another one is to prevent the possible defects to be introduced into the software products by people - it is done mainly through various automated and bidirectional traceabilities.

7. Conclusions

This paper described a new revolutionary software engineering process model based on complexity science – the NSE process model, where almost all software engineering tasks/activities are performed holistically and globally. Preliminary applications show that compared with the existing linear process models it is possible for NSE to help users double their productivity and halve their cost, and remove 99% to 99.99 defects in their software products.

8. References
