TiD2016
质量竞争力大会
软件研发顶级盛会

下一代软件研发
SOFTWARE DEVELOPMENT
更高 更新 更深

基于复杂性科学的NSE非线性与定量软件工程体系

熊继光

艾赛软件科技有限公司 伯文软件科技有限公司

目　录

1、何谓软件？

现有软件的定义为：软件包括1) 一个计算机程序，它的执行提供所需的特性、功能与性能； 2) 数据结构，使得该计算机程序可以恰当地处理信息； 3) 描述操作与使用该程序的文档（Pressman，《软件工程》）。

这样定义的软件割裂了文档与源码间的有机联系而不具备双向可追溯性，丢弃了在软件开发过程中通过动态与静态软件质量度量所建立的度量结果与测试结果数据库，没有使用这样的数据库的在线智能代理（高度自动化与相当程度智能化工具）“伴随”而使得软件不具备很好的适应性和可维护性。

1、何谓软件？（续）

基于NSE方法的软件的定义为：1) 一个计算机程序，它的执行提供所需的特性、功能与性能； 2) 数据结构，使得该计算机程序可以恰当地处理信息； 3) 描述操作与使用该程序的文档 - 它们与源码可以相互追溯而且始终保持一致； 4）在软件开发过程中进行静态与动态质量度量生成的数据库（包括测试用例和在测试过程中俘获的测试操作）；5)并提供委托开发方（通过免费的NSE体系实施支撑平台的软件产品验收工具包）的一系列在线智能代理（高度自动化和相当程度的智能化的工具集）来利用上述四个方面的资源来帮助用户进行软件的自动化验收以及需求实现的自动验证与确认，并有效底处理软件的复杂性和解决软件的可靠性、可变性、一致性、可追溯性与可维护性。

2、何谓一个科技领域的革命？

根据被广泛接受的“科学革命的结构”一书（作者 Kuh T, 1962）的严格定义，一个科技领域的革命必须满足三大条件：

（1）该领域的体系（体制）的根本性转移；

（2）能有效地解决该领域存在的一系列根本性问题；

（3）所提供的有效解决现存一系列根本性问题的方案具有唯一性。

这一简介将介绍一个革命性的软件工程体系 -- 非线性、整体性与定量软件工程体系。

3、软件工程面临的困境

（1）质量往往不达标甚至项目失败

（2）生产率低

（3）开发成本高

（4）难以进行产品的修改维护 -- 在需求变更的实现过程中往往引起副作用

现有软件工程存在的问题

（1）基于还原论：认为系统的整体就等于其局部之和

（2）采用了线性单向增量、迭代模型

（3）采用从顶向下的线性与定性软件开发方法，几乎都不支持软件维护

（4）现有不可执行、不可动态测试的软件建模

（5）测试体系存在的问题，一是时机不对，二是知其然不知其所以然

（6）现有软件质量保证体系存在问题，是重在事后而不是事前解决问题

（7）现有软件维护体系存在的问题，文档与源代码分割、没有双向可追溯性机制支持下进行的

4、NSE体系产生与发展背景

1、20世纪九十年代初在美国创立国际软件自动化公司，从事软件测试和质量管理，研制的产品**Hindsight**是当时业界最好的产品，**被太阳微系统、波音公司等企业使用。其中太阳微系统选为除了操作系统之外的软件测试工具；波音公司选用是因为该产品支持RTCA-DO178B-LevelA MC/DC测试覆盖率要求。**

2、二十世纪九十年代中旬，第二代: Panorama（“全景”），被《软件工程 实践者的方法》作者Pressman誉为“提供了面向对象软件开发完整的工具集”。**用户包括： SUN Microsystems, IBM Japan, HP/HP Japan, SONY, NTT, 富士通， 三菱， Nikon，CANON，西门子，东芝等许多大公司。**

4、NSE体系产生与发展背景（续）

3、进入二十一世纪，推出了Panorama++（“全景++”）/艾赛银弹,在国内中国船舶重工集团公司第701研究所(又称中国舰船设计研究中心)、中国航空研究院602研究所、上海航天局809研究所等应用。

UNIX Today! Products

057 OCTOBER 29, 1990 The Newspaper Of Open Systems Computing A CMP Publication

Fixing C Code

BY PAUL KRILL

Santa Clara—Advanced Software Automation (ASA) has released its Hindsight software maintenance tool, designed to cut the time programmers spend fixing existing C code.

Through the use of interactive structure charts and active logic diagrams, ASA's Hindsight software maintenance environment speeds understanding of code modification effects on the rest of a program, according to ASA. The product also locates test coverage deficiencies and updates documentation directly from code.

Sun Microsystems has adopted Hindsight as a software development suite testing tool. Hindsight will be used to evaluate test suites on non-OS software, such as compilers, programming environments and network software, said Charles Miller, a Sun software technology engineer.

The product works on major Unix platforms, including Hewlett-Packard/Apollo, Sun, Digital Equipment and IBM. It is priced from $12,000, for the structure chart feature only, to $23,000 for the structure chart plus all modules, including structural complexity and test coverage. Shipments begin by mid-November.

Although Hindsight currently supports only C, ASA is working on COBOL, FORTRAN and C++ versions, with no release dates set.

ASA senior vice president Tom McHugh said the tool helps programmers understand code that might be obscure to anyone but the original programmer. "Once you understand the code, it helps you evaluate performance, [reliability] and quality," he said.

Operating under X-Window-based interfaces such as Motif and Open Look, Hindsight covers the spectrum of maintenance tasks without requiring new methodologies or source code changes, the company said. Hindsight's automatically generated structure chart provides a graphic overview of a program structure.

Hindsight can overlay bar graphs on each function in the structure chart to depict test coverage, run-time performance, module size and complexity data. Using a mouse, programmers, including engineers and managers, navigate through structure charts and diagrams, pop-up logic flow or complexity diagrams and select feature buttons.

"You can do in maybe two or three minutes with [Hindsight] what would take hours or days in terms of understanding how the code works," McHugh said.

Miller said Hindsight builds upon standard interfaces. The product uses interfaces similar to *tcove*, which is a standard Unix utility, and *prof*, a run-time profiler, to generate results, he said.

"We've evaluated multiple test coverage tools," such as Verilog Logiscope and TCAT, from Software Research, Miller said. "Hindsight seems to provide the best graphical interface [and] the most usable graphical interface, as well as an integrated environment."

The product also features assisted code tracing, allowing users to follow function calls and references by pointing to a reference, then jump to the referenced line. This ability is included in Hindsight's block-flow logic diagram, the J-Diagram, a graphic representation of both high-level and detailed procedural logic, according to ASA.

Documentation updates also are generated after a change is made to the source code. More than 20 tables and charts document function cross-references, size, compactness, complexity, test coverage, run-time performance and other traits.

ASA can be reached at 408-492-1668.

NSE体系建立的理论体系之一

关于非线性软件工程体系的两本专著

NSE体系建立的理论体系之二

三篇被选入“2013 Recent Advances in Computer Science”一书的论文：

（1）Nonlinear and Quantitative Software Engineering Method Based on Complexity Science

（2）Transparent-Box Method Combining Structural and Functional Software Testing together Seamlessly

（3）Automated Generation of Software Documents Consistent with and Traceable to and from Source Code

国际软件工程大师对NSE体系和支撑平台的评价

From Professor C. Y. Ramamoorthy：

“...NSE 非线性软件工程体系的诞生正是其时！我相信，NSE体系不仅打破了软件开发的瓶颈，而且还有可能在相关的组织间形成新的非线性与定量软件工程产业链,...”

From Professor R. T. Yeh:"我很赞赏你使用复杂性科学来解决软件问题。。。我相信，你的工作已经提供了一些证据，说明这将产生很大的成果。恭喜！..."

From Professor Kunii: “... NSE体系是一个标志：表明线性的、部分的、局部的与定性的软件工程的时代已经成为过去；非线性的、整体的、全局性与定量的软件工程的时代已经来临！...”

NSE体系已经被软件行业智联联盟选为新的软件工程标准，并指定熊继光负责组织力量详细制定

NSE体系被指定为我国软件行业智联联盟新制定的软件工程标准

2015年7月14日下午2时，在有2000多软件精英参加的TID2015质量竞争力大会上，按照工信部要求成立的“智联标准委员会”宣布：在成功制定了软件质量与成本核算的标准基础上，今年要制定四项新的标准，包括“基于复杂性科学的NSE非线性与定量软件工程体系”--这是由我们创立的、具有完整自主知识产权的革命性软件工程体系。

智联联盟标委会成立仪式

我被任命为基于复杂性科学的NSE软件工程标准组长时起身致谢

NSE体系的实施支撑平台通过了专家鉴定会的鉴定

NSE体系的实施支撑平台Panorama（全景）++/艾赛银弹
通过了软件行业智联联盟组织的专家鉴定会的鉴定

【智联IT好创新】NSE产品成功通过产品鉴定

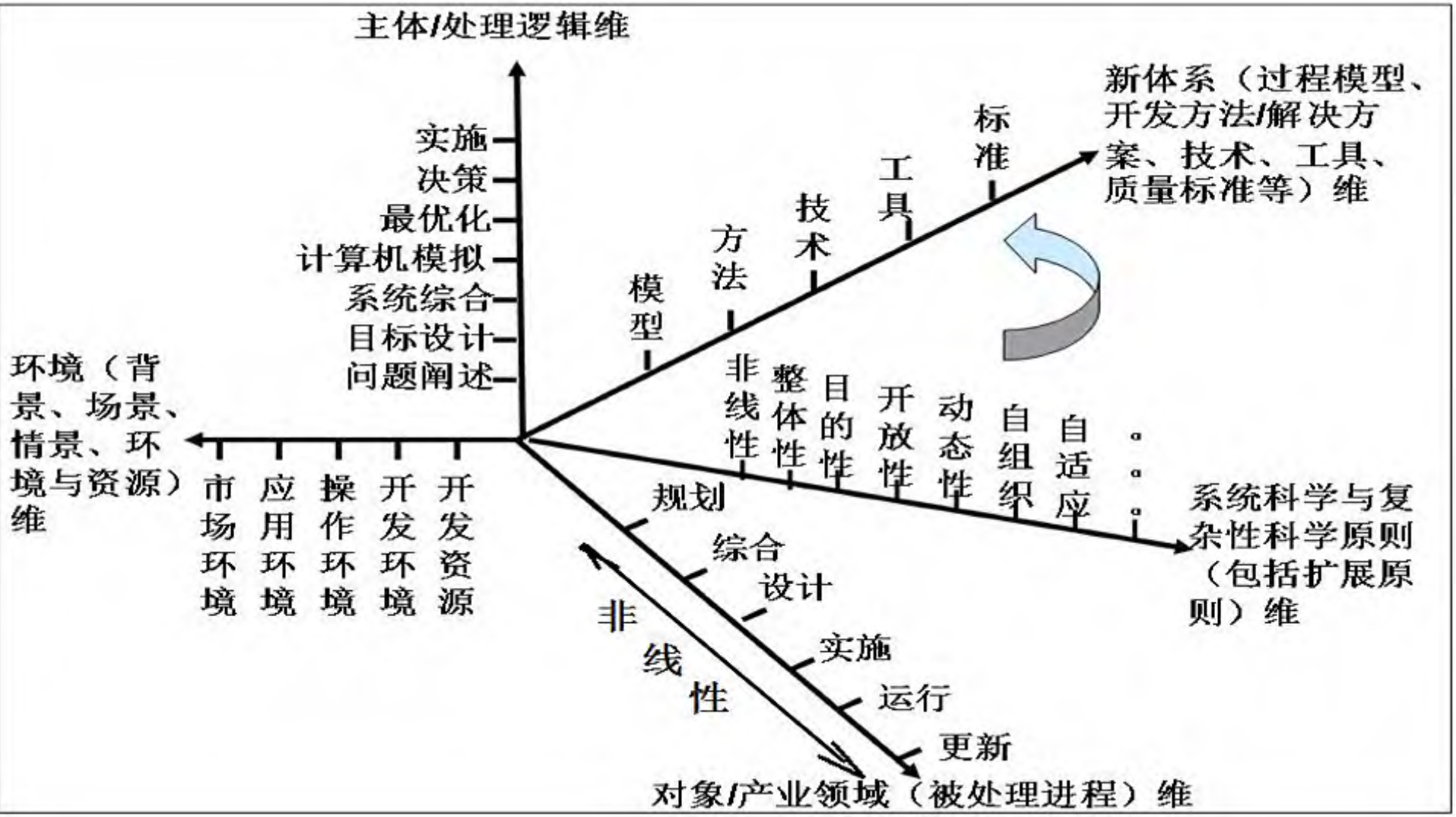
2015-08-31 智联联盟

2015年8月24日，由中关村智联软件服务业质量创新联盟组织召开的“IT好创新——新一代软件工程体系与应用NSE”产品鉴定会在北京召开，NSE产品成功通过IT好创新产品鉴定。

5、NSE体系的构成

- 非线性与定量软件开发方法，
- 所基于的一系列创新技术
- 透明盒软件测试体系
- 非线性与整体性和可执行软件建模体系
- 基于缺陷预防的软件质量保证体系
- 自动化文档体系
- 非线性、整体性和定量软件维护体系
- 软件开发过程融为一体的项目管理体系
- NSE体系的实施支撑平台
- 层次化可增量更新的大数据处理体系

基于复杂性科学的NSE体系的创立工作框架

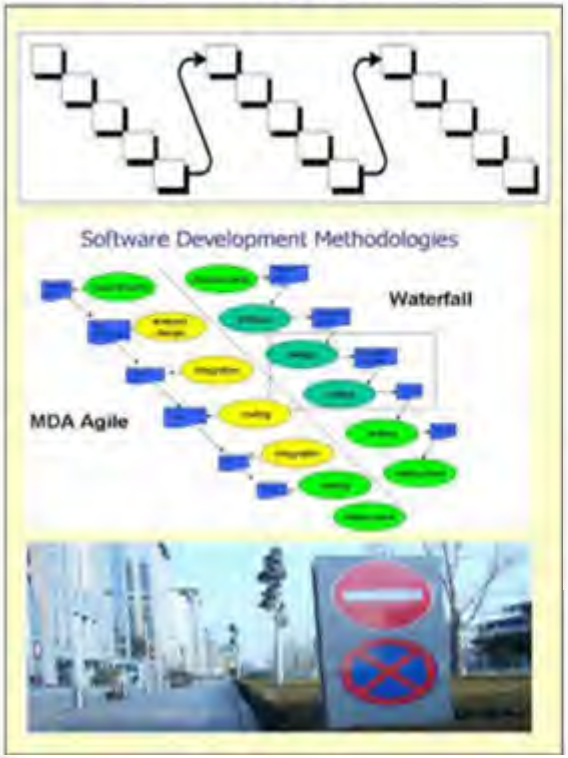


6、NSE方法与传统方法的总体比较

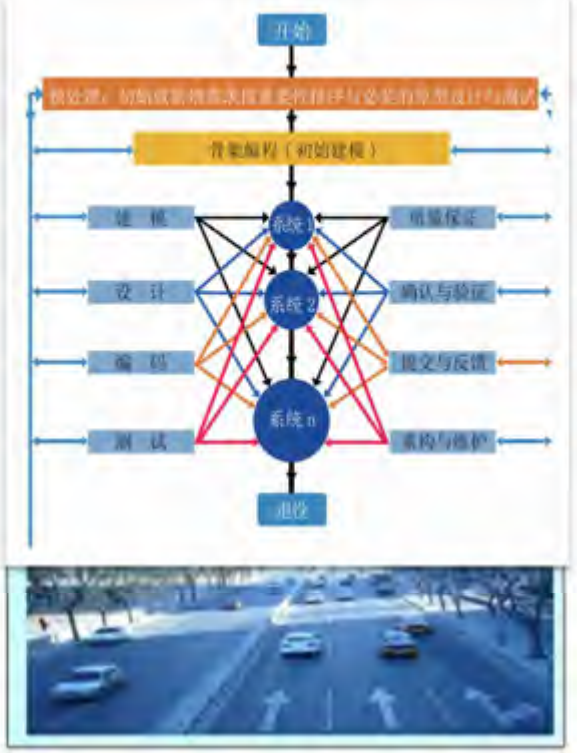
NSE方法	比较项	传统软件开发方法
01 复杂性科学	理论基础	01 还原论
02 非线性与定量	开发方法	02 线性与定性
03 基于缺陷预防、全过程动态测试、采用可追溯文档与源码的审议与可视化	质量保证	03 基于编码后的动态测试和软件审议
04 自动生成	文档生成	04 主要用 手工制作
05 全过程及其工作产品	可视化	05 主要用于建模
06 非线性、整体、定量进行	软件维护	06 线性、局部、定性进行
07 与软件开发过程融为一体，可相互追溯，及时发现和解决问题	项目管理	07 与软件开发过程分离，难于及时发现和解决问题

软件开发过程的比较

传统的软件开发方法



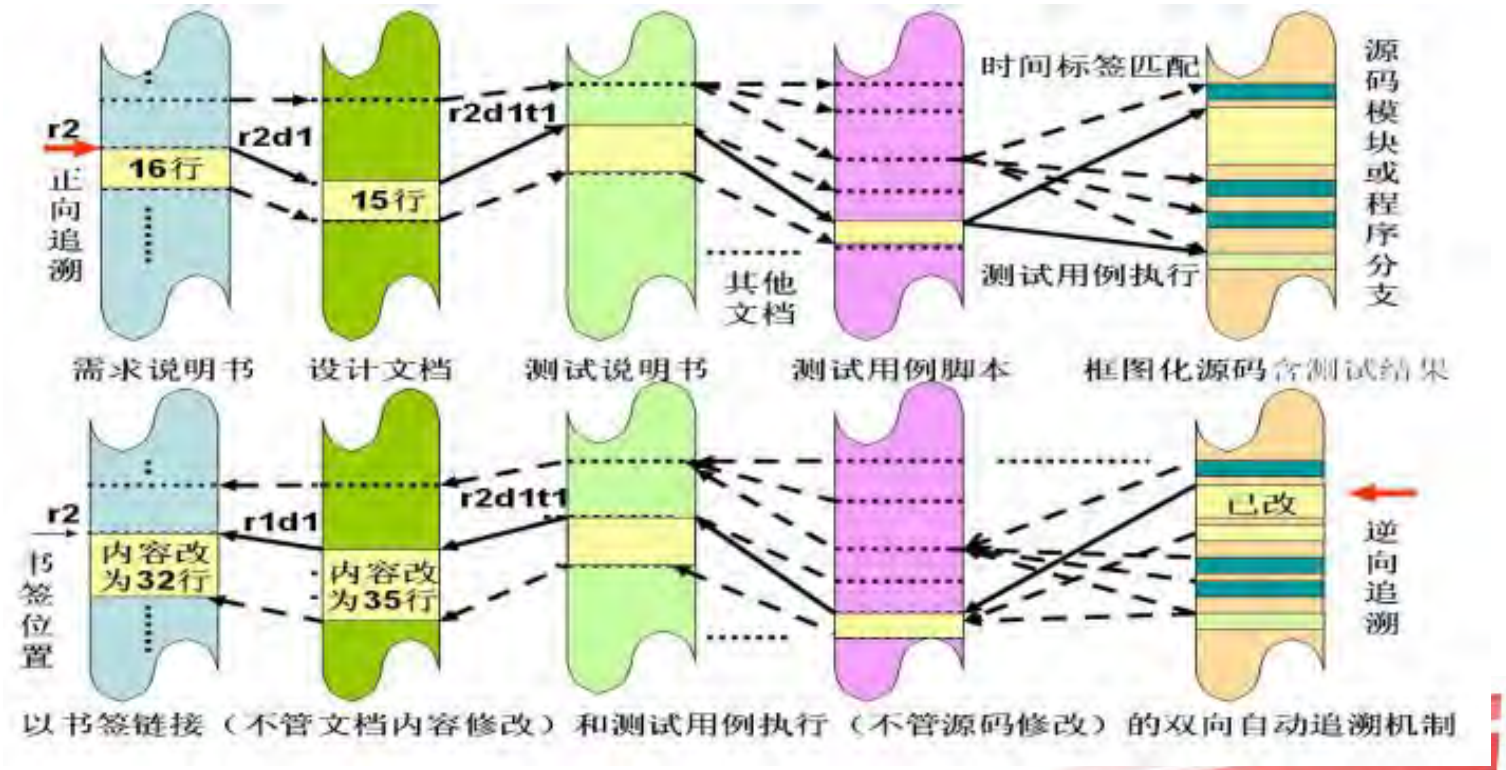
NSE方法



1、传统的螺旋模型和增量开发模型都是单向单轨的开发方式。。

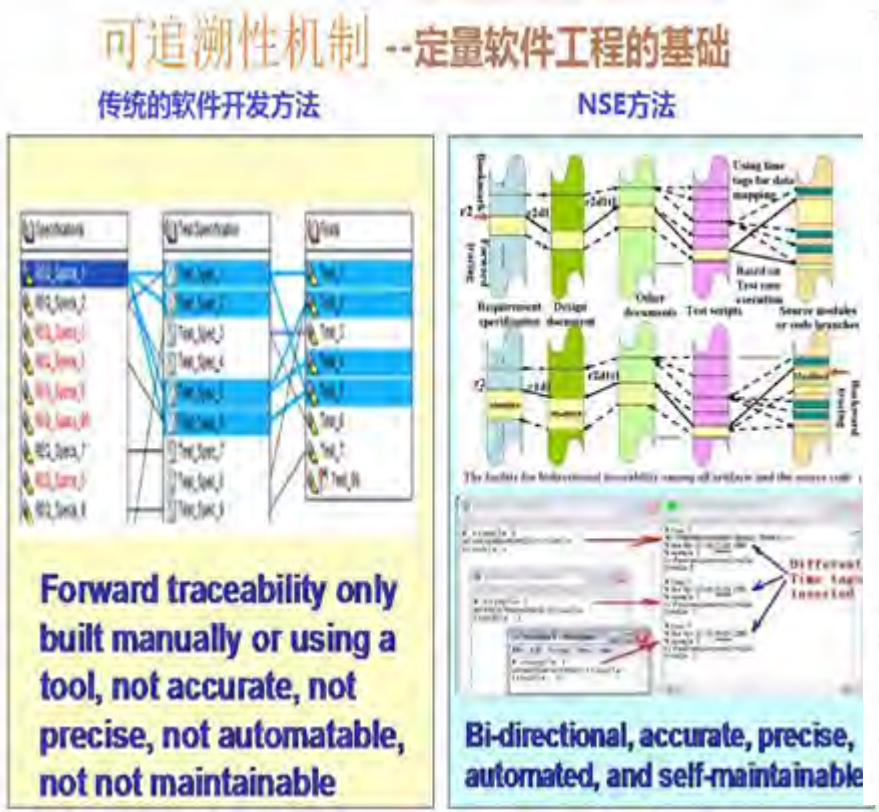
2、 NSE是非线性、增量式、多轨、并行、实时、双向迭代的，并将软件开发与维护融为一体、软件开发与需求管理融为一体、实现全过程缺陷预防、支撑团队实时沟通、高度适应需求的变更。

NSE双向可追溯性机制



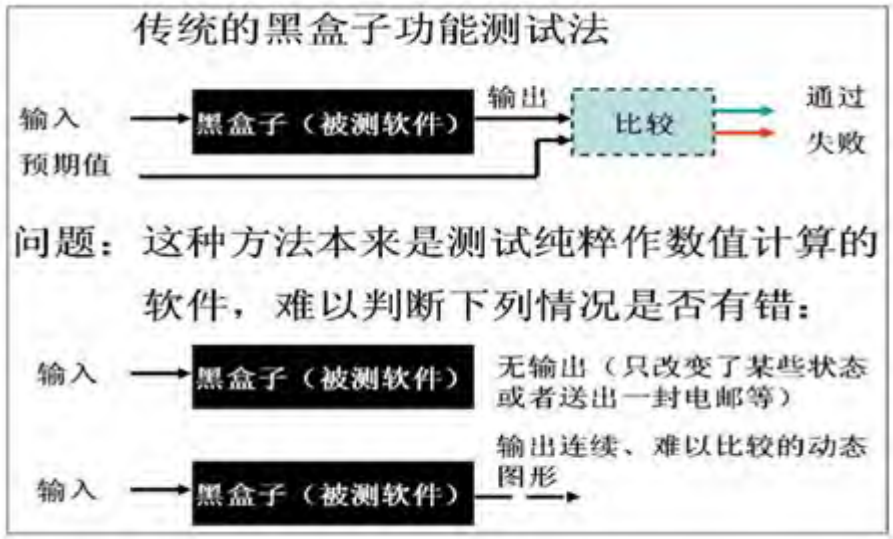
在需求说明书、设计文档与其他文档、测试用例与源码之间的双向可追溯性机制

可追溯性的比较



比较项	传统可追溯性技术	NSE可追溯性技术
是动态还是静态可追溯性技术?	静态	动态
可自动建立与自维护?	否	是
单向还是双向?	单向	双向
准确?	否（往往与源码不一致）	是（与源码一致）
精确?	否（仅可到模块级）	是（精确到语句级）

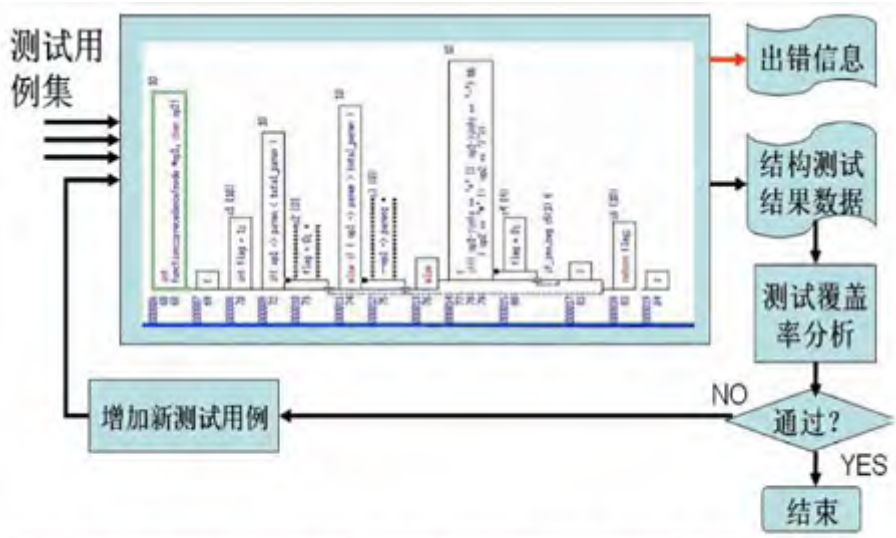
软件测试---传统的黑盒测试



1、功能测试与结构测试分离，往往是知其然而不知其所以然

2、需要在完成编码后才能动态地应用 -- 太晚了

软件测试---传统的结构测试



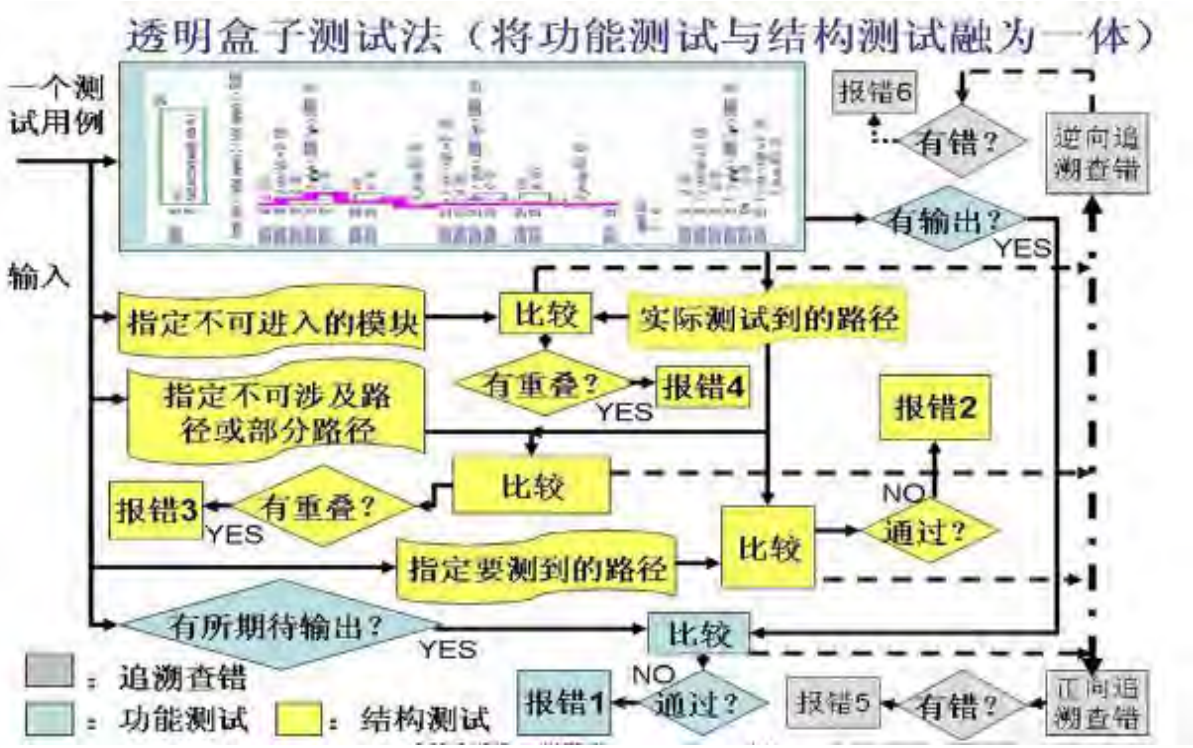
问题：所测试的，是已有的软件，不管是不是所需要的软件

只能测试“已有”的软件，不管是不是“所需”的软件（不管是否满足需求）

灰盒子测试法

这种方法可以找到比只用功能测试法或者只用结构测试法更多的软件错误。但它只是两个方法的、机械式的叠加，依然不能适应未来高质量的软件的测试要求

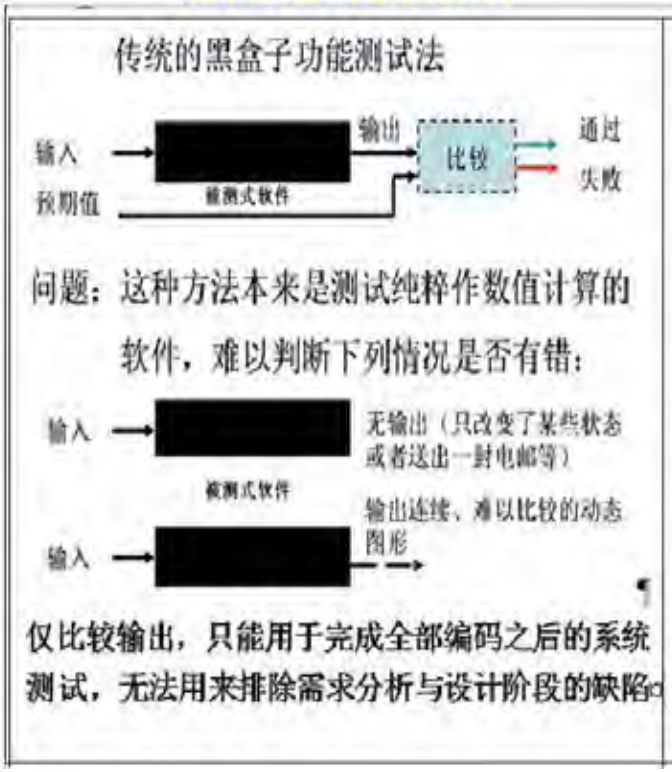
NSE软件测试方法---透明盒测试法



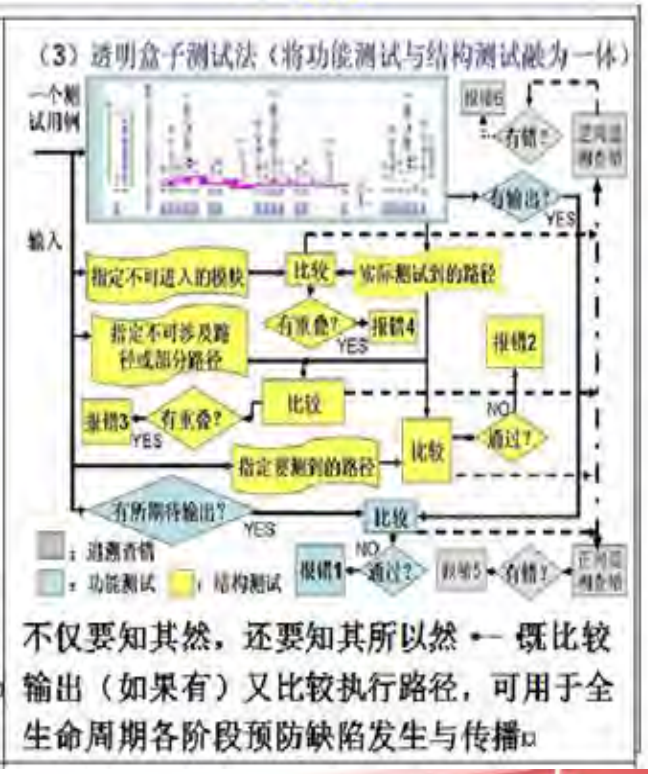
将软件结构测试与功能测试无缝地结合在一起：对于一组测试输入，不仅帮助用户检查被测试软件的输出（如果有，可以无，特别是在建模与设计阶段，通常都没有有实际有效的程序处理输出），看看它是否与所期待的结果一致，还帮助用户检查被处理软件的实际执行路径，看看是否与所期待的执行路径一致（或者，当其执行路径非常复杂时，是否包含了其中关键性的一段路径）。这特别适合于软件建模阶段和设计阶段的动态测试，因为这时所期待的执行路径都比较简单，容易设定（用程序控制流程设定）。

软件测试的比较

传统的软件开发方法



NSE方法

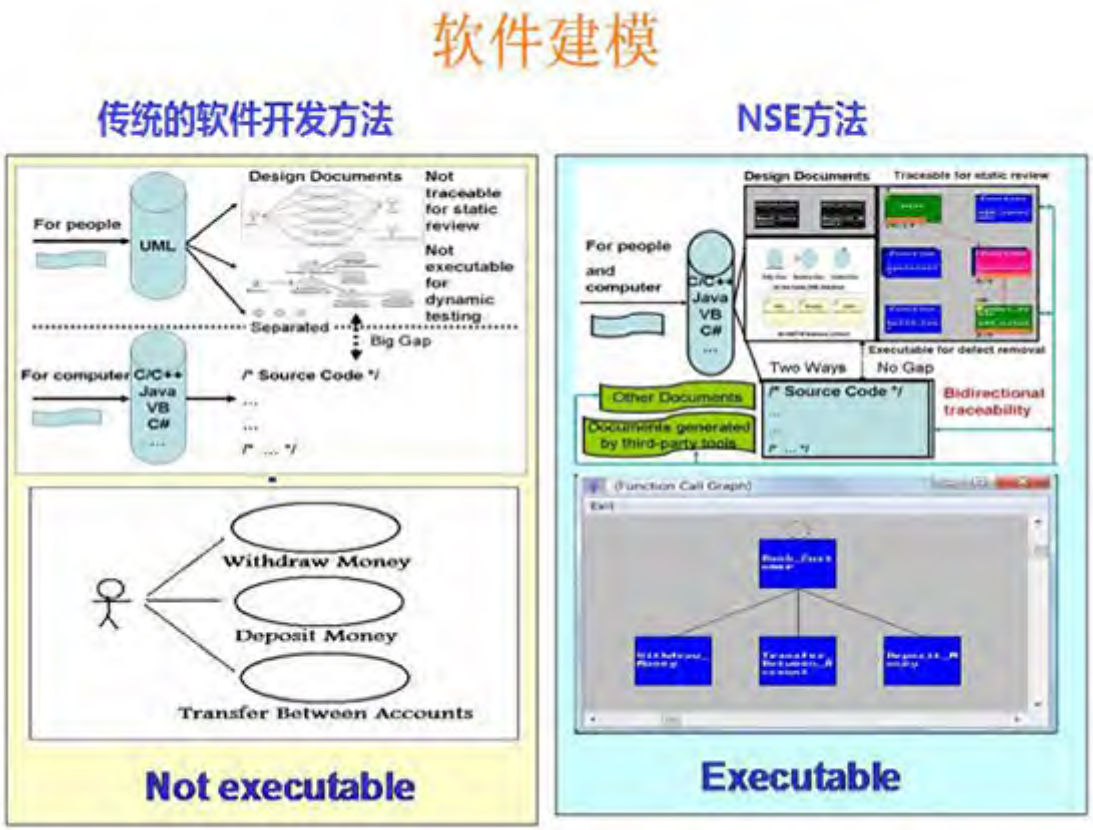


1、传统的软件工程体系主要采用黑盒功能测试方法（在完成编码之后进行）和白盒结构测试方法（在完成各个程序单元的编码之后进行），外加性能测试，压力测试，加载测试等。

2、NSE方法所采用的是独创的透明盒测试方法、它将功能测试与结构测试融为一体，不仅检查输出还检查执行路径。

3、NSE自动建立测试用例与被测试模块和程序分支之间的、准确而且精确的双向可追溯性。

建模技术的比较



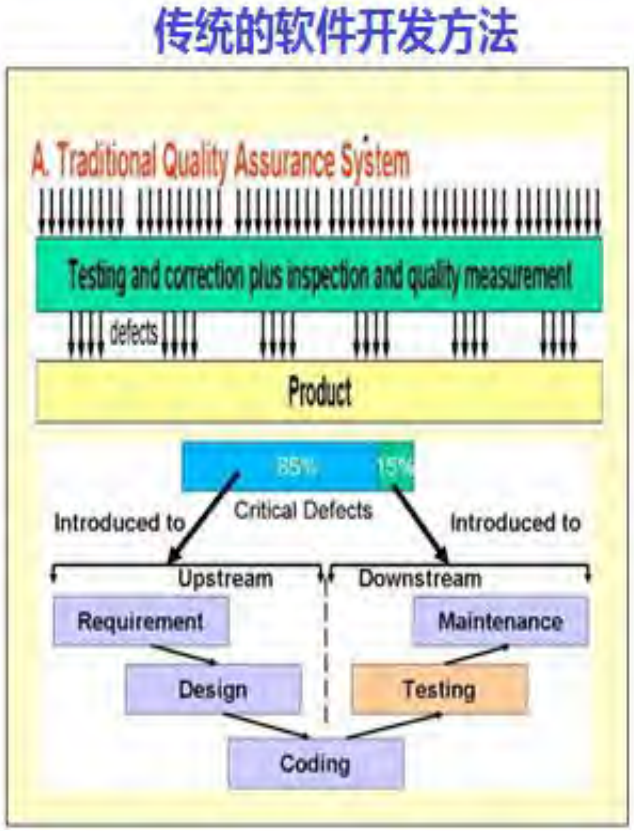
传统建模：

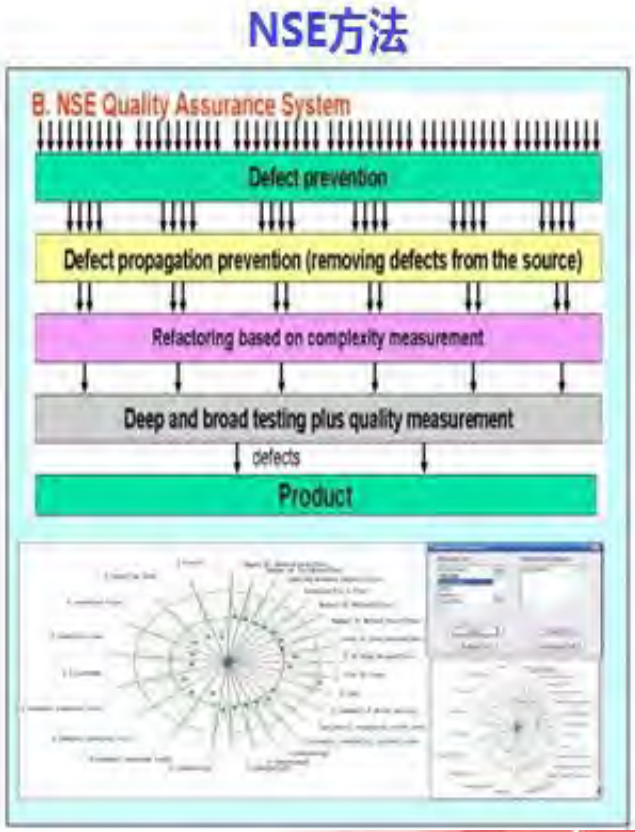
（1）难以自动化；
（2）非整体性；
（3）缺乏可追溯性；
（4）不可执行；
（5）造成建模与实现分别用不同的语言而几乎无法达到一致

NSE建模：

支持用同一语言进行软件建模和软件实现使得建模成为初始实现，实现成为进一步建模， 用于供人们理解一个软件系统的诸多彩色图形都是通过构架编程自动生成的，具有自动化、整体性、可追溯性，高一致性和可以动态执行的优点

软件质量保证体系方面的比较

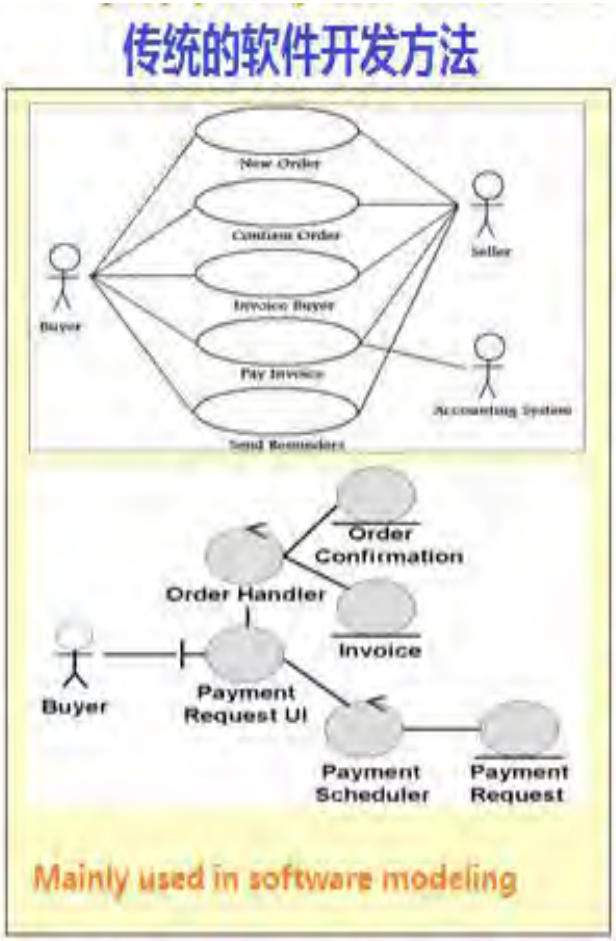


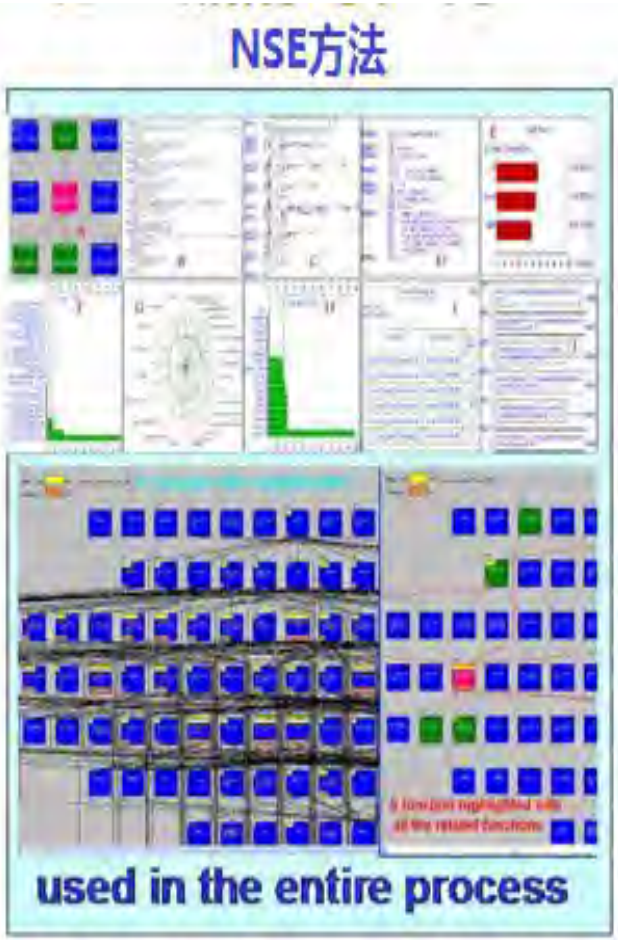


1、传统的软件工程体系的质量保证方法，主要靠编码后的测试以及静态软件审议，但85%左右的关键性软件错误都是在建模阶段和设计阶段引入的。

2、NSE非线性与定量软件工程体系的质量保证，采取以防为主以治为辅的方法，首先是软件缺陷的主动预防， 然后是已经引入系统的缺陷的传播预防,加上复杂性高的模块的重构，以及深度和广度的软件测试与质量度量。

软件开发方法过程及工作产品可视化的比较



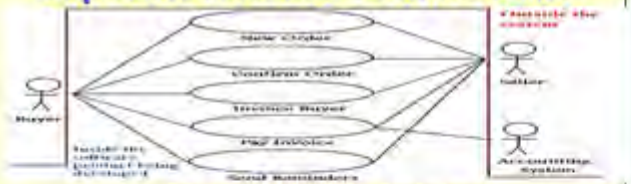


传统的软件工程体系往往仅仅在建模阶段可视化；而NSE非线性与定量软件工程体系则可实现软件开发与维护的全过程及其工作产品的可视化。

软件文档体系的比较

传统的软件工程方法

Separated to the source code



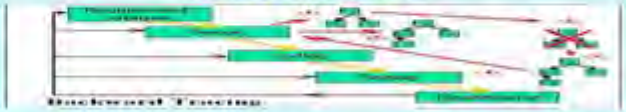
Not automatable

Inconsistent with and not traceable to and from the code

NSE方法

Combined with source code

Automatable



Consistent with and traceable to and from the source code

传统的软件文档体系存在一系列缺点：

（1）分立，难以自动生成
（2）彼此难以实现自动链接
（3）无法与源码进行双向追溯
（4）往往与修改过的源码不一致

NSE自动化文档体系有NSE如下特点：

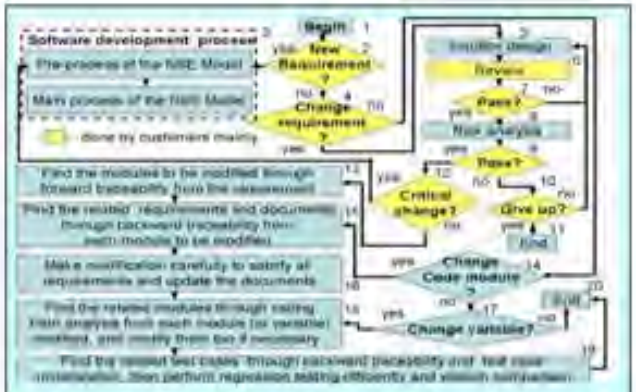
（1）结构化， 自动生成
（2）彼此间自动或以书签为节点链接
（3）可与源码进行双向追溯
（4）可始终与源码保持一致
（5）可与源码进行一体化管理和维护

软件维护体系的比较

传统的软件开发方法

No software maintenance process model defined; responding to software changes blindly, partially, locally, and qualitatively.

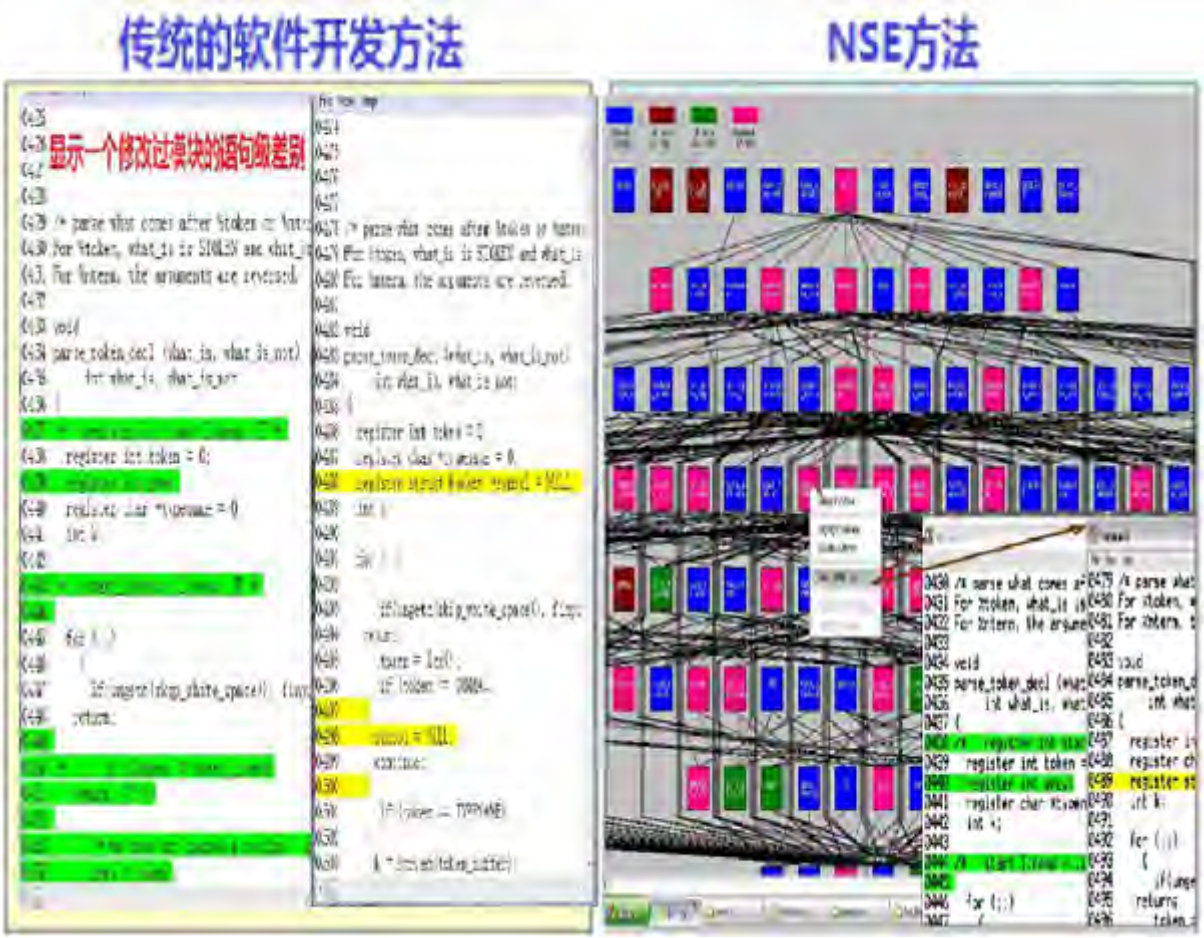
NSE方法



Responding to software changes holistically, globally, and quantitatively with side-effect prevention through bi-directional traceability

传统的软件工程体系的软件维护是线性地、局部地、定性地进行的， 往往改正一个错误就有20-50%的机会引入一个新的错误，造成被维护的软件越来越不稳定； NSE非线性与定量软件工程体系的软件维护，是非线性地、整体性地、全局地与定量地进行的，而且通过种种双向可追溯性预防软件修改副作用的发生，以确保软件系统能长期稳定工作

软件版本比较方法与技术的比较



传统的软件版本比较方法和技术，往往采用局部的、文件级或者语句级的版本比较方法，很难让用户了解一个软件产品的不同版本之间的差别的全貌，以及已经修改过的版本可能会引起哪些模块也可能要作相应的修改 -- 缺乏可追溯性。不同的是，NSE方法所采用的，是整体性全局性多极比较方法和技术，并附有可追溯性机制来帮助用户掌握一个修改过的模块可以影响到什么模块 -- 它们也许应该做相应的修改。

7、 NSE体系支撑平台架构与产品

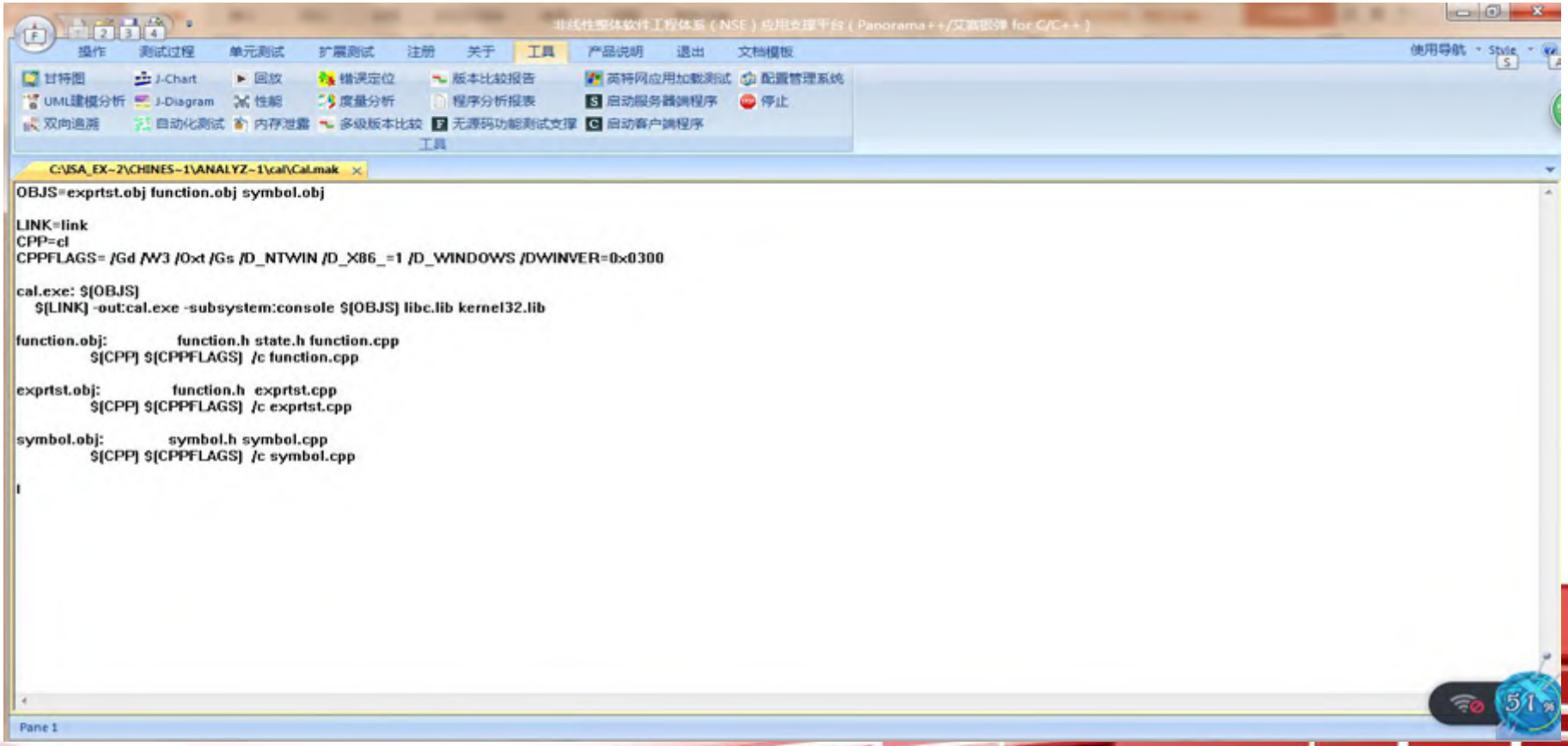
进度管理 UML建模 CVS
软件建模 软件设计 软件测试（功能测试、性能测试） 内存泄漏 版本比较
需求确认 软件度量 SQA 文档生成 软件维护

基于NSE体系的统一平台（ 统一格式 ）

C C++ Java VB APL

Windows Linux UNIX

8、NSE产品演示


操作
测试过程
单元测试
扩展测试
注册
关于
工具
产品说明
退出
文档模板
甘特图
J-Chart
回放
错误定位
版本比较报告
英特网应用加载测试
配置管理系统
UML建模分析
J-Diagram
性能
度量分析
程序分析报表
启动服务器端程序
停止
双向追踪
自动化测试
内存泄露
多级版本比较
无源码功能测试支撑
启动客户端程序
工具
C:\ISA_EX~2\CHINES~1\ANALYZ~1\cal\Cal.mak
OBJS=exprtst.obj function.obj symbol.obj
LINK=link
CPP=cl
CPPFLAGS= /Gd /W3 /Ox /Gs /D_NTWIN /D_X86_=1 /D_WINDOWS /DWINVER=0x0300
cal.exe: ${OBJS}
${LINK} -out:cal.exe -subsystem:console ${OBJS} libc.lib kernel32.lib
function.obj: function.h state.h function.cpp
${CPP} ${CPPFLAGS} /c function.cpp
exprtst.obj: function.h exprtst.cpp
${CPP} ${CPPFLAGS} /c exprtst.cpp
symbol.obj: symbol.h symbol.cpp
${CPP} ${CPPFLAGS} /c symbol.cpp
Pane 1

9、追求敏捷中的敏捷 - NSE Vs. Agile

1. 持续、尽早交付有价值的软件以满足客户
2. 拥抱需求变更，甚至是在开发的后期。
3. 频繁交付可执行的软件，从几周到几个月，交付时间越短越好。
4. 在整个项目过程中，业务人员和开发人员必须每天在一起工作。
5. 激发每个团队成员的积极性来打造项目。为他们提供所需支持
6. 在一个开发团队内部最有效的传递信息的方式是面对面的交流。
7. 可执行的软件是进度的首要检验对象。
8. 赞助商，开发人员和用户应该尽可能保持一致的步伐。
9. 保持关注先进的技术与设计会增强敏捷性。
10. 尽量用艺术化来简单阐述未完成的工作是很有必要的。
11. 最好的架构，需求，和设计出自于自我组织管理的团队。
12. 每隔一段时间，回顾反思如何让团队变得更高效，并相应地调整其行为。 如何实施，如何实施得更好更敏捷？

问与答

谢谢！