Configuring the Software Development Process on Linux

Dr. Adam Kolawa

***Para**Soft* Corporation

ak@parasoft.com

Why Modify Your Development Process?

- Current industry trends are forcing companies to turn out applications in 2-3 months
- If you don’t have an effective development process, you will not survive

Goals of This Presentation

- Demonstrate the necessity of configuring a development process that effectively prevents errors
- Describe what such a development process should look like
- Discuss how tools fit into this development process
  - *Describe what commercial and public domain tools help you perform each step or assess readiness to progress,*
  - *Explain how to effectively configure public domain tools*

ParaSoft

Error Prevention

- Errors are not inevitable
- Errors occur because of bad coding and development practices
- Examples in C++ and Java
- Preventing errors results in a higher-quality product than introducing errors, then trying to find and remove them

ParaSoft®

Example 1

IBM Study

"Projects that aim from the beginning at achieving the shortest possible schedules regardless of quality considerations tend to have fairly high frequencies of both schedule and cost overruns. Software projects that aim initially at achieving the highest possible levels of quality and reliability tend to have the best schedule adherence records, the highest productivity, and even the best marketplace success."*

* Jones, Capers. Programming Productivity. New York: McGraw-Hill, 1986.

ParaSoft

Example 2

- W. Edwards Deming: Use statistical quality control to build quality into product-- implement a process that prevents errors, rather than focus on the product itself (Total Quality Control)
- In the 1970s, American auto manufacturers' failure to focus on process and error prevention prevented them from improving product quality
  - *They assumed defects were inevitable and did not try to stop causing defects*
  - *Instead, they tried to remove existing defects*
  - *Manufacturers failed to significantly reduce the number of product defects*

Development Process

- Customers
- Development teams
- Methodology
- Practices
- Tools

ParaSoft

Methodologies

- Waterfall
- Spiral
- RAD
- eXtreme programming

ParaSoft

Practices

- Certain practices are universally acknowledged as beneficial, regardless of development methodology
- Code construction
  - *Coding standards, code reviews, unit and application testing*
  - *Debugging support: firewalls, debugging information and debugging methods*
- Source monitoring
  - *Source control, automated and regular builds, bug tracking, automatic error detection, metrics*

Coding Standards

- Standard code formatting
- Standard coding constructs
  - *Beyond syntactically permissible*
  - *Standardized practices*
- Coding standards promote…
  - *A reduced chance of introducing defects*
  - *A reduced chance that defects will go unnoticed during code review*
  - *Enhanced communication through the code*
  - *Increased efficiency*

ParaSoft®

Source Control

- Central repository for the entire source base
- Provide a record of the evolution of the source base
  - *Who, when, why*
- Reduce the risk of change
  - *Recovering older, more stable version*
  - *Cheaper to try different approaches*
- Facilitate group access to a common source base
  - *Network, remote access*
- Source tagging
  - *Named collection of sources*

ParaSoft

Automated Builds

- Minimize the overhead in assembling the application pieces
- Identify certain classes of interface errors
- Promote frequent builds of the entire application
- Provide the basis for automated regular testing at the application level

ParaSoft®

Unit Testing

- Write the test *before* the code!
- Use the tests to document the behavior of the object
- Guaranteed maintenance of the test suite along with the code
- Thoroughly test the entire publicly accessible interface
- Test the boundary conditions
  - *Out of range arguments, typos, NULL objects, numbers that are too big or too small*

ParaSoft

Unit Testing - Continued

- Unit testing provides…
  - *Increased confidence that the unit is performing as expected*
  - *Continuous assessment of the quality of the code*
  - *Early detection of incompatible changes*
- Helps minimize the risk of change by providing a means of verifying correctness and detecting fault

ParaSoft

Application Testing

- Functional tests document the expected application behavior
- Necessary for assessing…
  - *Overall application behavior*
  - *Unit integration*
  - *Performance*
  - *Resource requirements*

ParaSoft

Regression Tests

- Non-trivial applications have non-local defects
- Repairing a defect sometimes disturbs a sizeable portion of the system
- Regression tests…
  - *Provide a warning system against non-local defects*
  - *Shield against poor understanding of application behavior by the maintainer*
- A defect is not repaired until a test case that detects its presence is placed in the regression test suite

ParaSoft

Additional Testing

- In-house testing
- Beta cycles
- Release candidates

ParaSoft

Bug Tracking

- Ensure appropriate people are notified of problems
- Facilitate bug reporting
- Correlate bugs to source versions
- Facilitate tracking of individual problems
- Defect counts and defect rates are fundamental quality metrics

ParaSoft

Code Reviews

- Promote collective ownership of the code
- Distribute the knowledge about the implementation details to the team
- Allow more experienced developers to share know-how
- Most importantly, code reviews are a very effective way to detect defects

ParaSoft®

Debugging Support

- Add logic for pre-conditions and post-conditions
- Retain the code fragments used to exercise the code during debugging sessions
- Write routines specifically for use during debugging
  - *Expensive data structure validations*
  - *Multi-object interrelations*
  - *Detailed examination of memory owned by an object*

ParaSoft

Automatic Error Detection

- Simplest way to detect defects
- Works well with automatic application builds and test suites
- Some tools can generate test cases automatically

ParaSoft

Coverage Analysis

- Measures the fraction of the application code that has been exercised
- Uncovers certain types of defects
- Provides a metric of the thoroughness of test suites

ParaSoft®

Development Tools for Linux

- Free
- Commercial

ParaSoft

Version Control

- CVS
  - *Download from www.cyclic.com*
- Present in most Linux distributions
- Reads and writes a standard file format

GNATS

- Download from http://sources.redhat.com/
- Installation requirements:
  - *Root access*
  - *GNU m4 installed*
- Easily customizable
- Integration with emacs

ParaSoft

Bugzilla

- Download from http://www.mozilla.org/projects/bugzilla/
- Web-enabled

ParaSoft

Unit Testing Tools

Types of tools available:

- Integrated
- Task-specific
- Perform black-box, white-box and/or regression testing
- Various degrees of automation

ParaSoft

Unit Testing Tools

Look at:

- *Ease of use*
- *Degree of automation*
  - *Builds harness automatically*
  - *Creates test cases automatically*
  - *Automatically generates stubs*
- *Customizability*
- *Variety of tests performed*
- *Coverage of automatic tests*

ParaSoft

Automatic Runtime Error Detection Tools

Look at:

- *Technology that drives the error detection*
- *Number and types of errors found*
- *Ability to detect errors in threads*
- *Precision of error messages*
- *Memory areas in which it finds errors*

ParaSoft

ElectricFence

- Available in most Linux distributions
- Detects two kinds of dynamic memory misuses:
  - *Out-of-bounds accesses*
  - *Accesses to de-allocated memory*

ParaSoft

Memory Profiling Tools

Look at:

- *Types of reports available*
- *Real-time profiling*

ParaSoft

Coverage Analysis Tools

Look at:

- *Testing method (by block or by line)*
- *How it works with your other tools*

ParaSoft

Conclusions

- Survival requires a development process that promotes shipping reliable applications as rapidly as possible
- Tools– both freely available and commercial– are critical to maintaining such a development process
- As the confidence of tool vendors in Linux increases, more specialized development tools should become available