

An Epistle on the Importance of Software Metrics

David Bernholdt
ORNL

Why?

- People considering adopting CCA want to know
 - what it is going to “cost” them
 - And what they will gain from it
 - *Also, our reviewers stressed the need to try to quantify the costs and benefits*
 - *Quantitative measurements may help us identify problem areas*
- Costs CCA include
 - Time and effort for development
 - Performance overheads
- Benefits of CCA include
 - Reuse of components in multiple applications
 - Ease of building up application from (mostly?) existing components

Software Metrics

- A lot of people put a lot of effort into trying to define meaningful metrics...
- ...but it is not clear if any have succeeded
- See, for example:
 - http://irb.cs.uni-magdeburg.de/sw-eng/us/bibliography/bib_main.shtml
 - (Thanks, Randy)
- So for this purpose, I propose some very simple, easy to capture measurements...

Performance Metrics

- Show CCA *maintains* performance
 - This is not a terribly positive measure. How to make it sound better?
- Measure performance before and after CCAifying
 - Must be careful to have valid comparison (no algorithmic changes)
- Look at overheads on method invocation
 - Already some measurements out there (ANL, ORNL+LLNL)
 - Need more – *every new application*
- Look at overheads on parallelism
 - Shouldn't be any, but we need to demonstrate on BIG problems!
- Show CCA *improving* performance through swapping interoperable components

“Development” Metrics

- *We can't do controlled studies, so we mostly have to look at converting existing code to CCA*
- Number of lines “touched”
 - Assumes original code is “reasonable”
- Lines of code in component
 - Compared to lines for same functionality in original
- Time required (hours of effort) to CCAify

More Subjective Measures

- Time required to do something new in CCA
 - Compared to estimate of time using “traditional” methods
- “Overhead” of developing common interfaces
 - Case 1: Interface would have been designed w/o CCA
 - Did CCA make it harder or easier? Why?
 - Case 2: Interface would *not* have been designed w/o CCA
 - Why not? What about CCA makes it feasible? Or is it just good timing?
 - How do you measure the value of having a common interfaces?

Benefits are Hard to Quantify

- Software “reuse” currently happens via cut-and-paste
 - Does CCA make that better? How much?
- How do you define “reuse”?
 - What constitutes a distinct application a component might be reused in?
 - What fraction of components should we *expect* to be reused?
 - How much should we expect a given component to be reused?

My Charge to the Congregation...

- When working on components,
 - Track time spent
 - Track code changes (keep a copy of original)
 - Do performance comparisons (apples to apples)
 - Think about/record how the CCA experience development experience subjectively compares to traditional experience
- Evangelize users to follow the same path
- Think about better ways to assess CCA

While I've Got the Pulpit...

- A new CCA Overview paper is under development
 - Update to 1999 HPDC paper
- Tentative target: Int'l J High-Perf. Comp. Appls.
 - Jack Dongarra promises timely review & publication
- Basic outline and introductory material are under development, partial versions in CVS
- Authorship will be based on actual contributions (CVS check-in)
- Contact me to participate
- CVSROOT cca-forum.org:/cvs/papers
- Module papers
- Path papers/2003/cca-overview