Excellent Code:
Taking Authorship or following rules.

Go to ask tom and search for Ten Commandments

Then search for Anthony to see my response

Cosmin: Need Help With Top Ten

cosmin -- Thanks for the question regarding "The 10 (Oracle Development) Commandments", version 9.2
Submitted on 21-Nov-2005 13:34 Eastern US time
Last updated 12-Dec-2005 14:17

You Asked

hi Tom,

I've been tasked w/ raising awareness in my organization

(through a top 10 "commandments" -- to do list)

about better ways to write applications, for speed and development

purposes as well.

I know that Feuerestein has a similar list although a lot more geared towards PL/SQL.

I wanted to get your feedback, particularly on probably consolidating some of these points and adding some others that you feel are appropriate and most common in the enterprise.

information architects
designing enduring quality

Cosmin: 10 Commandments

1. Thou shalt comment all thy code.
2. Thou shalt use proper indexes.
3. Thou shalt maximize sql and minimize pl/sql.
4. Thou shalt instrument thy code.
5. Thou shalt benchmark thy code.
6. Thou shalt debug code efficiently.
7. Thou shalt stress test w/ significant data.
8. Thou shalt make use of bulk processing.
9. Thou shalt minimize client code and maximize server code.
10. Thou shalt review code with peers.

information architects
designing enduring quality

Cosmin: Looking For Help

I know that most of these require one or more chapters or lectures/etc -- each in its own right -- I'm not debating that, I am merely providing a starting point from which discussions & enterprise Oracle development can ensue.

Yes, this is a developer's list. I know we can probably come up with an Architect's list (IOT, partitioning, compression, etc etc -- too many to list) and probably a DBA one as well :-) -- is this something of interest, to put in an future Appendix list?

Yes, I also know that for each rule there are exceptions and for each exeception there are further exceptions but hey, that's what makes for wonderful magic (hint-hint, a famous recent Oracle book introduction) :-)

thx much,
Cosmin

information architects
designing enduring quality

Anthony: Is this the right question?

Writing Quality Code: Following commandments or responsibly authoring it
December 11, 2005 - 9pm US/Eastern
Reviewer: **Anthony Harper** from Orlando, Florida

- These 'commandments' seem to be actually too specific, in so doing they may miss the mark if the purpose is to help the organization raise awareness about writing better code.

- Since most of the follow up to this list ended up as a discussion of the merits of comments and an interpretation of a 'school' of thought perhaps these 'commandments' did fail to raise awareness of how to write good, maintainable code for the current requirements and for future programmers to maintain.

information architects
designing enduring quality

Anthony: Authoring Is Simple

In thinking about this issue to help the organization that I am working at currently (and the future developers at that company) I always return to that old slogan 'THINK'.....and my interpretation of it:

- Think about how the code should be written to fulfill the business requirements you have been given.
- Write the code to fulfill your design.
- Review what you've written.
- Refactor what you've written.
- Review the code to see if it actually functions as you expected it to.
- Repeat the process from the beginning, using your first attempt as a draft.

information architects
designing enduring quality

Anthony: Reviewing Is Easy

This could also be simplified as the following:

- Write code that clearly illustrates the business process.
- Review and refactor the code until it actually performs as expected.
- Review, revise and refactor the code until it is clear enough to have another developer understand it without additional verbal explanation.
- Too often programmers don't take the time to review their own code from the top to the bottom.
- By the time you have finished a particular piece of code you will have a new perspective from which to review it.
- This new perspective in turn will help you to refactor the code to make it more clear and more reusable.

Anthony: Save Future Time

- If we take the time to 'author' the code, we take on a responsibility to review, revise, refactor and edit the code.
- This time is significantly less than the amount of time it will take a future developer to rewrite the code in order to understand it for the purposes of making changes.
- If we take the time to review and revise the code when first writing it, we will have added comments, structure and explanations as needed to let the text 'speak for itself', or else we couldn't quite follow it during review.

information architects
designing enduring quality

Anthony: Authors KISS Code

- If you are extremely dedicated to the 'Profession' of writing quality code: review, revise and refactor the code again until it is clear enough for a manager (ie, non programmer) to understand it without further ado.

- Part of the beauty of Oracle SQL and Oracle PL/SQL is the fact that it is written in clear English.

- This can be leveraged to make the code comprehensible to a non technical audience....if you can do this then you can 'kiss' the code goodbye, and leave it to fulfill its purpose until someone needs to revisit it again when its original purposes have changed.

information architects
designing enduring quality

Anthony: Develop Responsibly

- I'm not sure we need to have 'commandments' that must be dogmatically adhered to, or 'schools of thought' (i.e. XP) that use a currently popular style or metaphor to drive our development processes.

- I am sure that we all need to take the extra time to 'author' the code so that it is something to be proud of, and so that it is something that could serve as a textbook example of the business process that the code has been written to carry out.

- The 'ROI' on this extra editing and review time up front will always outweigh the costs of future maintenance.

information architects
designing enduring quality

Anthony: Don't Miss Dinner

- Another successful strategy is to remind developers to write code that will be understandable and clear when debugging at 6:30 pm on Friday when there is a production problem.

- If the code is not clear, if it is not clearly structured and broken into unit testable components, you will not get out of the office until 9 pm.

- When the heat is on and you are stepping through the code, you will thank yourself and your predecessors for taking the time to make the code clear and easy to understand.

PS. I did review this letter three times and I am sure I have missed something and it will not be entirely clear to everyone. Please help to make it a better review with your feedback.

information architects
designing enduring quality

Tom: Nice Advice For Experts

Followup December 12, 2005 - 7am US/Eastern:

I THINK sometimes bulleted lists as above are useful.

tho shalt INSTRUMENT your code.

for example - a directive.

Your list is good - for the expert. For the novice, what does "write the code to fulfill your design" mean exactly?

If you are the novice, or just the average/below average - to what end will reviewing your code be?

If you haven't instrumented it, commented it, benchmarked it, stressed it, tested it, etc.

If you don't know bulk processing is "good", and you review your slow by slow code - what will you detect other than the code must be good?

So, I think I still for the most part like this particular list. Could it be expanded - sure, almost infinitely large probably and at various levels of "dirction".

Anthony's Summary

The perspective of responsible authoring for code is the first advice I would give to any developer, regardless of skill.

This perspective is also the final thought that I would like to leave everyone with.

information architects
designing enduring quality