CONCLUSION

Testing is a crucial element of the maintenance phase of software. The more software you already have developed, the harder it is to ensure that bug fixes and enhancements don't cause additional headaches. Good software development technique can minimize the likelihood that a code change will have unexpected repercussions, but can't reduce the chances to zero. Testing should be frequent and thorough, which means using as much automation as possible.

Unit testing is becoming increasingly common, and is even being picked up by development phase programmers in the form of Extreme Programming. GUI testing is harder, and you'll have to come up with policies that balance the required frequency and thoroughness with your own sanity.

A well-designed test bed and test set will save you a lot of apologies to your customers, who don't enjoy updating software and certainly don't want to do it more than once to fix any particular problem. Using maintenance testing techniques can save time, money, and dignity.

REFERENCES

- *"ISO/IEC 14764:2006 Software Engineering — Software Life Cycle Processes — Maintenance". Iso.org. 2011-12-17. Retrieved 2013-12-02.*
- ^ *Soleimani Neysiani, Behzad; Babamir, Seyed Morteza; Aritsugi, Masayoshi (2020-10-01). "Efficient feature extraction model for validation performance improvement of duplicate bug report detection in software bug triage systems". Information and Software Technology.* ***126****: 106344. doi:10.1016/j.infsof.2020.106344. S2CID 219733047.*
- ^ Pigoski, Thomas M., 1997: Practical software maintenance: Best practices for managing your software investment. Wiley Computer Pub. (New York)
- ^ Eick, S., Graves, T., Karr, A., Marron, J., and Mockus, A. 2001. Does Code Decay? Assessing Evidence from Change Management Data. IEEE Transactions on Software Engineering. 27(1) 1-12.