Using Time Series Models for Defect Prediction in Software Release Planning

James Tunnell
Central Washington University
Computational Science Program

March 27, 2015

Outline

1. Introduction
    - Release Planning
    - Quality Control
    - Defect Prediction
2. Motivation
    - Release Plan Optimization
    - Explanatory Model
3. Related Work
    - Code Analysis
    - Historical Information
4. Time Series Modeling
    - Time Series
    - Autoregressive Models
    - Endogeneity and Exogeneity
5. References

Introduction

Release Planning Objectives

- Two primary objectives of software release planning are:
  - Improving functionality
  - Maintaining quality
- Both of these objectives are constrained by limits on development time and cost.

Quality Control

- Software defects (bugs) are inevitable
- Sufficient time should be available to ensure good quality (by testing and bug-fixing)
- Otherwise, there is a risk of
    - Low quality (failure to meet objective)
    - Schedule slip (failure to respect constraint)
- This quality control (QC) time can be allowed for by limiting the scope of work in the planned release

Quality Control (cont'd)

- To support release planning, QC time can be estimated
- Assumption: QC time depends (at least partly) on the number of software defects introduced
- Then, a basis for estimating QC time would be the predicted number of defects

Defect Prediction

- Approaches to defect prediction tend to focus on either
  - Code analysis
    - Lines of code
    - Number of decisions
    - Code churn
  - Historical information
    - Regression analysis
    - Time series modeling
- A multivariate time series model with exogenous inputs was chosen

Motivation

Release Plan Optimization

- A release plan is formed by selecting features and improvements to work on
- Release plans can be compared by the expected revenue they will generate
- This optimization problem is posed as The Next Release Problem (NRP)

Release Plan Optimization (cont'd)

- The NRP is an abstract optimization problem
- In practice, QC time should be considered to ensure constraints are respected
- With the help of a defect prediction model, QC time can be estimated
- In this context, release plans are being compared
- For a defect prediction model to be useful, it should depend in some way on the basic elements of the release plan (planned new features and improvements)

Explanatory Model

- Assumption: the number of defects in the future depends on more than just the number of defects in the past
- A defect prediction model that depends only on previous numbers of defects is not explanatory
- Such a non-explanatory model would always predict the same number of defects

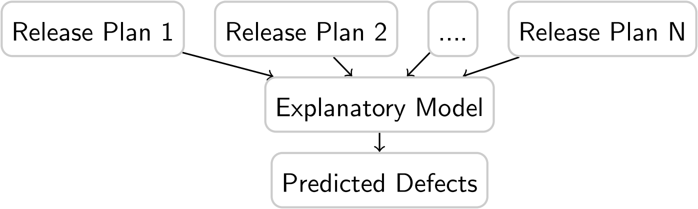

Figure : A non-explanatory model.

Explanatory Model (cont'd)

- A model could also depend on the key factors of a release plan
- This would be an explanatory model structure
- Such a model can potentially predict a different number of defects for every release plan

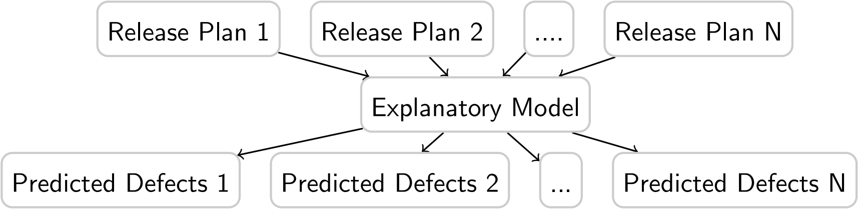

Figure : An explanatory model.

Related Work

Defect Prediction using Code Analysis

- Approaches using code analysis:
  - Akiyama used lines of code (LOC), number of decisions, and the number of subroutine calls [1]
  - Gafney also used LOC [2]
  - Henry and Kafura use information taken from design documents [3]
  - Nagappan and Ball use relative code churn (lines modified) [5]
- These approaches all depend on specific design or implementation information
- This information is not available at the release planning stage

Defect Prediction using Historical Information

- Approaches using historical information:
  - Li et al. extrapolate parameters of a regression model [4]
  - Singh et al. use an ARIMA time series model [6]
- Both approaches are non-specific to design or implementation
- However, neither approach is explanatory

Time Series Modeling

Time Series

- A time series is a collection of observations that occur in order
- The process underlying a time series is assumed to be stochastic (non-deterministic)
- Each observation might depend on one or more previous observations
- This dependence is termed autocorrelation

Autoregressive Models

- A basic autoregressive (AR) model is a linear combination of previous values
- A white noise term accounts for stochastic fluctuation
- An $AR(p)$ model for predicting a value X at time t is

$$X_t = c + \sum_{i=1}^{p} \phi_t X_{t-1} + \epsilon_t \tag{1}$$

where $\phi_1, \phi_2, ..., \phi_p$ are the p parameters, c is a constant, and ϵ_t is the white noise term

Autoregressive Models (cont'd)

- Extending the AR model to be multivariate results in a Vector AR (VAR) model
- This model can support time series for defect count, improvements, and new features

Endogeneity and Exogeneity

- Under a VAR model, the behavior of each time series is explained by both its own past values and the past values of the other time series
- This makes the variables endogenous
- An alternative is that a time series is only used to explain other time series
- This type of explanatory variable is called exogenous, and could be considered an input
- Exogenouse variables are not explained by the model

Endogeneity and Exogeneity (cont'd)

- The desired model does not need to explain features and improvements
- Instead, these are used to explain defects
- Planned features and improvements can be made exogenous
- By also considering exogenous variables, a VAR model would become a VARX model

References I

F. Akiyama.
An example of software system debugging.
In *IFIP Congress (1)*, volume 71, pages 353–359, 1971.

J. E. Gaffney.
Estimating the number of faults in code.
Software Engineering, IEEE Transactions on, SE-10(4):459–464, July 1984.

S. Henry and D. Kafura.
The evaluation of software systems' structure using quantitative software metrics.
Software: Practice and Experience, 14(6):561–573, 1984.

References II

- P. L. Li, M. Shaw, J. Herbsleb, B. Ray, and P. Santhanam.
  Empirical evaluation of defect projection models for widely-deployed production software systems.
  SIGSOFT Softw. Eng. Notes, 29(6):263–272, Oct. 2004.

- N. Nagappan and T. Ball.
  Use of relative code churn measures to predict system defect density.
  In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 284–292. IEEE, 2005.

- L. L. Singh, A. M. Abbas, F. Ahmad, and S. Ramaswamy.
  Predicting software bugs using arima model.
  In *Proceedings of the 48th Annual Southeast Regional Conference*, page 27. ACM, 2010.