Software Development Profile Model

Arya Khoshkhou [a,b], Michel Cukier [a,c], Ali Mosleh [a]

[a] Center for Risk & Reliability, University of Maryland
[b] Lockheed Martin Corporation
[c] The Institute for Systems Research

14th Annual Practical Software & Systems Measurement
User's Group Conference
July 26-30, 2010
New Orleans, Louisiana

Background

- Despite hundreds of software reliability models, software engineering discipline is still struggling to establish a common framework for software reliability estimation
- Some software reliability models use failure data for their parameter estimation
- Other models rely on historical data from similar projects for parameter estimation
- As the result many models can't be used during the early phases of software development process
- Often people and process execution are ignored in software reliability models

Software's Distinct Characteristics

- From the reliability point of view software has two distinct characteristics

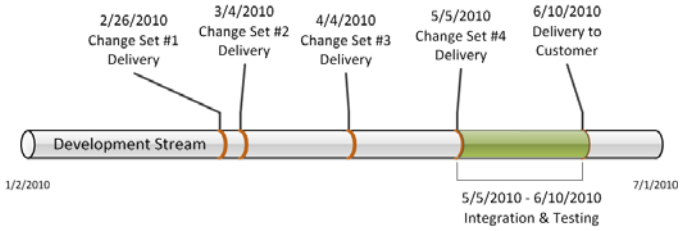
1. Software is subject to continuous change and enhancements
   - Software changes requires human interaction which can't be easily automated
   - The software development process relies heavily on human judgment
   - All software defects are caused by human error
2. Software failures have some deterministic properties
   - Software does not age (it does not fail due to random failure)
   - Once a software fault is removed it will never fail for the same reason again
   - The random nature of software failure is due to the unknown locations of faults in the software program

Motivation

- The motivation behind this work was to introduce a causal model for estimating the defect proneness of various software artifacts based on software development characteristics
- Empirical studies show a strong correlation between software change history and defect proneness of software artifacts

Software Development Process

- Software is developed overtime and software changes are maintained in a developed stream
- A development stream is a collection of all development activities occurred in chronological order
- After software changes are made, they are inspected, tested and the delivered back to the stream in the form of a change set



DEFINITION

- A ***software construct*** is the smallest software unit for which data is being collected (software line of code, function point, source statement, etc.)
- ***Software artifacts*** are all the products that are created during software development containing software constructs (source file, module, class, documentation, etc.)

Software Development Profile Model

- Software Development Profile Model is based on the assumption that when a software construct (SLOC, module, function point, source statement, etc.) is touched within a change set 'c', there is a chance that it can become defective.
- Let's define

$$Z^c = \begin{cases} 1, if\ a\ construct\ did\ not\ become\ defective\ in\ change\ set\ 'c' \\ 0, if\ a\ construct\ became\ defective\ in\ change\ set\ 'c' \end{cases}$$

Software Development Profile Model

- Since we don't know which constructs became defective in the change set 'c', Z^c is a random variable.
- Let's define r^c as the change set reliability, which is the probability that a given construct is correctly modified (created) during change set 'c'

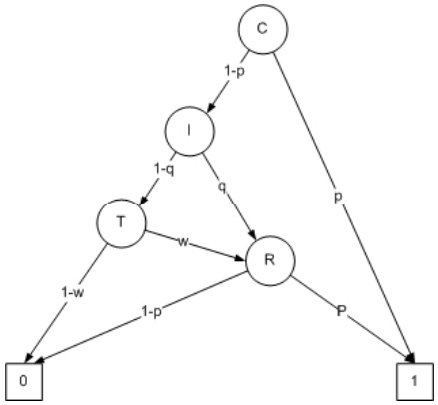
$$r^c = \Pr\{Z^c = 1\}$$

SDPM – Estimating r^c

- To estimate r^c, we use a Binary Decision Diagram (BDD) to capture the activities that occur during each change set
- Each decision node in the BDD represents an activity during the change set
- Each edge from a node represents the assignment of a success or failure of the activity

SDPM - Modeling Software Development Activities

- Example of a simple Binary Decision Diagram
  - Nodes:
    - C = Coding
    - I = Inspection
    - T = String /SWIT Testing
    - R = Repair (Re-work)
  - Edges
    - P = coding reliability
    - q = inspection reliability
    - w = testing reliability



SDPM – Modeling Change Set Reliability

- Based on the sample Binary Decision Diagram (BDD), the probability that a construct did not become defective during change set 'c' is

$$r^c = \Pr(Z^c = 1) = p^c + (1 - p^c) \cdot q^c \cdot p^c + (1 - p^c)(1 - q^c) \cdot w^c \cdot p^c$$

- We denote r^c as the change set reliability and p, q, w are the model parameters

SDPM – Parameter Estimation

- Let's define N^c as the estimated number of constructs that became defective in the change set 'c'
- Let's also define S^c is the size of the change set 'c' (number of constructs created or modified)
- and $\overline{p}^c$ as the coding unreliability of change set c
- Assuming the constructs become defective independently, we can write:

$$\Pr(N^c \mid S^c, \overline{p}^c) = \binom{S^c}{N^c} \cdot (\overline{p}^c)^{N^c} \cdot (1 - \overline{p}^c)^{S^c - N^c}$$

SDPM – Parameter Estimation

- N^c can be estimated during the inspection process of each change set using methods such as capture-recapture method
- Bayesian theorem can be used to obtain information about $\bar{p}^c$ based on N^c and size of the change set S^c
- We use the Beta distribution in conjunction with the binomial distribution to describe $\bar{p}^c$

$$B(\bar{p}^c \mid S^c, N^c) = \frac{\Gamma(S^c + 2)}{\Gamma(N^c + 1)\cdot\Gamma(S^c - N^c + 1)} \cdot (\bar{p}^c)^{N^c} \cdot (p^c)^{S^c - N^c}$$

SDPM – Parameter Estimation

- The posterior mean of the Beta distribution is

$$\hat{p}^c \approx 1 - \frac{N^c}{S^c} \quad for\ large\ N^c\ and\ S^c$$

- Similarly we can estimate the other model parameters using Bayesian theorem
- The posterior mean values of q and w are:

$$\hat{q}^c \approx \frac{i^c}{N^c} \quad and \quad \hat{w}^c \approx \frac{t^c}{(N-i)^c}$$

- Were 'i' is the number of defective constructs observed during inspection 't' is the number of defective constructs observed during SWIT testing

SDPM – Chance Set Reliability

- The change set reliability can then be expressed by:

$$\hat{r}^c = \Pr\{z^c = 1\} = \left(1 - \frac{N^c}{S^c}\right) \cdot \left(1 + \frac{D^c}{S^c}\right)$$

- S^c is the size of the change set c
- D^c is the total number of defective constructs observed during inspection and testing of the change set c
- N^c is the estimated number of defective constructs

SDPM – Modeling Change History

- Software is developed over a period of time using multiple change sets
- Software constructs can be modified within different change sets
- Each change set is influenced by different factors and will experience a different change set reliability
- It is necessary to capture the change history of software constructs

SDPM – Modeling Change History

- Configuration Management stores the information about the software change history
- This information can be obtained automatically using available tools
- Software change history can be stored in the Software Change Matrix SCM=($x_{i,c}$) where:

$$x_{i,c} = \begin{cases} r^c, & if\ the\ i-th\ construct\ was\ touched\ in\ change\ set\ c \\ 1, & otherwise \end{cases}$$

SDPM – Modeling Change History

- Example: The following SCM captures the change history of 9 constructs through 9 change sets.

$$SCM = \begin{pmatrix} 1 & 1 & r^3 & r^4 & 1 & 1 & r^7 & 1 & 1 \\ r^1 & 1 & 1 & 1 & 1 & 1 & 1 & r^8 & 1 \\ 1 & r^2 & 1 & 1 & r^5 & 1 & 1 & 1 & r^9 \\ 1 & 1 & 1 & r^4 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & r^7 & 1 & r^9 \\ 1 & 1 & r^3 & r^4 & r^5 & 1 & 1 & 1 & r^9 \\ r^1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ r^1 & 1 & 1 & 1 & 1 & r^6 & r^7 & r^8 & 1 \\ 1 & 1 & 1 & r^4 & r^5 & 1 & 1 & 1 & r^9 \end{pmatrix}$$

○ =Created

○ =Modified

SDPM – Modeling Change History

- Assuming the reliability of each change set is independent, then the reliability of the i-th construct can estimated by:

$$R_i^c = \prod_{j=1}^{c} x_{i,j}$$

- This means that the reliability of a given construct "i" is the product of the reliability of all change sets, during which it was touched
- If the construct is not touched in a change set, its reliability remains unchanged

SDPM – Estimating the Number and the Location of Remaining Defects

- The expected number of defective constructs in a given artifact (M) can be estimated by

$$E(M^c) = \sum_{i=1}^{m} \left[1 - R_i^c\right], \quad where\, m = |M|$$

Definition: Software Development Profile

- We define Software Development Profile (SDP) as the set of all the constructs in the development stream along with their reliabilities

$$SDP^{c} = \left\{ \left(i, R_i^c \right) \forall\, i \in SS, where\ R_i^c = \prod_{j=1}^{c} x_{i,j} \right\}$$

- Where:
  - 'c' is the index of the last change set
  - SS is the set of all constructs included in the Software Stream

Concluding Remarks

- Software Development Profile Model has some unique and interesting properties:
  - Flexible: SDPM can be applied to non-executable artifacts (text files, design documents, requirements, etc.)
  - Scalable: SDPM can be applied to the entire software solution or a subset of the project (specific function, critical modules, generated code, developed code, GUI code, etc.)
  - SDPM can be used to adjust software development activities to improve software reliability while the product is still in development

Questions & Comments

- Contact Information:
  - Email: arya.khoshkhou@lmco.com, aria@umd.edu
  - Office: 301-623-3536