Using System Dynamics to Optimize Software Testing Operations

Keen Sing Lee
Eduardo Malaga
Christos Sermpetis

Courtesy of Keen Sing Lee, Eduardo Malaga, and Christos Sermpetis. Used with permission.
Agenda

- Background
- Objective
- Model
- Simulation
- Results
- Conclusions
- Recommendations
Background

- Banco de Credito del Peru (BCP)
- ~ 100 “small scale” software projects per year
- Average project development time: 8 months, 4 developers, 2 testers
- Matrix PO Structure
  - Development Vs Testing Unit

Matrix Organization
- Project manager has tasking and budget authority
- Line manager has functional authority, promotions
- Team members remain in their functional organizations (have 2 bosses)
- Potential for conflicts
- Adv
  - PM is responsible
  - Resource flexibility
  - Continuity
  - Job security for team members
- Disadv
  - conflicts between functional (line) managers and PM
  - some aspects “fall through the cracks”

Banco de Credito del Peru:
- Established 100 years ago
- 5,000 of employees
- Largest bank in Peru
- Operations also in Bolivia, Bahamas, Colombia,

Main Banking Business applications (core) are bought.
Small scale developments around these applications - addition of new functionality, in order to:
  - create new interfaces in other organizations,
  - adapt to changes in legislation
  - to increase performance of the software infrastructure
  - Increase employee productivity
The figure above shows the work processes beginning from the feasibility study to production. A typical project takes 8 months with the development and testing phases accounting for 4.5 months. This is due to the rework cycles needed to correct the coding errors as illustrated by the simple DSM.

The team decided to focus on developing a system dynamics model with regards to these 2 phases. The objectives are to understand the testing cycle so as to minimize the cost and time for small projects that is specific to BCP’s current operations for small projects.

Total typical project time: 8 months
Breakdown in phases:
• Feasibility Study: 0.5 months
• Analysis: 2 months
• Development: 2.5 months
• Test & Corrective Development Iteration: 2 months
• Acceptance (Training & Installation): 0.5 months
• Production (First 2 weeks of real production): 0.5 months
How “testing” works:

- Testing cycle:
  - starts only when all errors of previous cycle corrected
  - ends when
    - max number of errors discovered OR
    - max amount of time allocated per cycle spent
- Probability of discovering errors decreases with decreasing error density

The figure on top zooms in on the development and testing phases.

When the developers finish coding, the batch is passed on for testing. During testing, when an error is detected the tester informs the developer so as to initiate rework.

Testing for that cycle will cease when either the maximum number of errors have been detected (eg. 30 per cycle) or when the maximum time per cycle has been reached (eg. 5 days).

The next testing cycle will only begin when all discovered errors from the previous cycle has been corrected.

It is expected that the probability of discovering errors decreases with decreasing error density as more testing cycles are performed.
Model Design

- Model development process
  - Structure (stock & flow diagram) to ensure consistency with BCP’s organization
  - Data to match BCP’s operational reality
- Basic Modeling assumptions:
  - Only one type of error (no differentiation between critical or non-critical errors)
  - Error correction time constant for all errors & cycles
  - All developers and testers equally skilled

The basic modeling assumptions are:

• First, 1 type of error. There is no differentiation between critical errors & non-critical errors. It is assumed that the initial cycle are tested for critical errors with cosmetic errors being tested later.

• Second, constant error correction time. Again an assumption given that the developers should be proficient with the core business applications.

• Third, developers and testers are equally skilled. We decided not to model the staff quality due to time constraints and our relative inexperience in system dynamics modeling.
The purpose of these stock and flow diagrams is to model how tasks flows from developers to testers. Upon error corrections the rework tasks shall be returned to the stock (tasks to be checked) for another testing cycle.

There is also a counter indicated by the stock (testing cycles) to count the number of rework cycles.

These Stock and Flow diagrams represent the tasks’ flow from the stock (tasks to be checked) by the developers to the stock (tasks checked) for testers via the checking rate. The checking rate is a function of the number of resources, checking finished, test time per cycle and tasks to be checked.

From the stock (tasks checked) it will go back to the stock (tasks to be checked) via the task submission rate which is a function of tasks checked and all errors solved.

The number of cycles repeated will be captured by the stock (testing cycles) through the new cycle rate which is a function of task submission, every time the tasks are passed from the stock (tasks checked) to the stock (tasks to be checked) it goes through the task submission rate which serves as a counter for the stock (testing cycles).

The auxiliary variable (test time per cycle) is a function of the table (testing speed) and the stock (testing cycles) that controls the checking rate.
The purpose of this stock and flow diagram is to model the error rework needed, starting from discovering the coding errors from undiscovered errors to reworking the errors and keep count of the errors committed (quality of coding team – bonus, etc).

****************************************

The Stock and Flow diagram starts with a source via the coding error rate which is a function of error introduction rate and error submission. It then passes to the stock (undiscovered errors) via the error discovery rate that is a function of the table (probability of detection), error density, checking rate, checking finished, undiscovered errors to the stock (discovered errors). It then passes to the stock (error solved) via the error rework rate which is a function of the number of development resources, rework rate, and discovered errors. From the stock (error solved) it goes to the stock (error inventory) via the error submission rate which is a function of the errors solved and all errors were solved.
..... modeled to capture the total time spent on the development and testing phases.

************************************************

The Stock and Flow diagram starts with a source via the time checking rate which is a function of checking rate and time step to the stock (cycle time). It then passes to the stock (total cycle time) via the checking end rate which is a function of the cycle time and checking finished.
This slide shows the complete system dynamics model. There are 8 exogenous inputs:

- Testing speed
- Testing resources
- Time step
- Maximum time per cycle
- Maximum errors per cycle
- Initial number of tasks to be checked
- Initial number of errors
- Error introduction rate
- Development resources
- Probability of detection

The value of these exogenous inputs were determined based on BCP’s data. From these, the team selected 4 inputs to perform the Design of Experiment.
The model considers four variables. These are the:

The number of testers
- The number of developers
- The maximum number of errors per cycle
- The maximum number of days per cycle

The last two variables are part of the current testing policy of the bank. Testing cycles are finished whenever the maximum number of errors or the maximum number of days are reached.

For each variable there were selected a range of possible values to be tested. In the slide are shown these values. Current values are also highlighted. Sensitivity analysis was performed changing each variables one at a time. 14 simulations were performed in total. In order to identify the optimum combination of values we measured for each simulation the following values:

- Total time spent on work: defined as the total days the developers and the testers were working.
- Work Days: defined as the total number of days required to complete all the testing cycles. It also takes into consideration the time when no testing is done and only the developers are working.
- Total MP Cost: defined as the total man power cost. This is calculated multiplying the number of work days by the number of people (testers and developers).
- Total Idle Cost: defined as the cost of time when the developers and tester were idle.
- Total Real MP Cost: defined as the cost of time when developers and testers were working.
- Ratio idle/total: Total idle cost/Total Cost
The following charts show the behavior of the model in terms of the costs (Total Cost, Real Cost and Idle Cost) and duration of the total testing phase (Work Days). Charts also show the current value used at the bank as well as the optimum value found in the simulations.

We observe that having more than 3 testers doesn’t reduce the duration of testing phase. Conversely, having just one tester increases the idle cost (i.e. developers waiting for errors to be fixed).
Increasing the number of developers always reduces the duration of the testing phase. However, lowest cost is achieved working with 6 developers.
Increasing the maximum number of days per cycle reduces both the duration of the testing phase and the overall MP cost.
Results

Analyzing the behavior of two the max number of errors and max number of days per cycle we observe that the optimal combination is 30 errors and 10 days. This seems to indicated that increasing the duration of cycling times could reduce the duration of the phase.
Recommendations

- Current values are close to optimum values yet there seems to be space for further improvement.
- Increasing the number of developers reduces the duration of the testing phase but increases the cost.
- 3 testers seems to be the ideal number for this project size.
- Increasing the maximum number of errors per cycle could decrease the Total MP Cost.
- Results are applicable for an average project in the bank, however, other specific factors must be taken into consideration.
Caveats

- Multi project environment not taken into consideration
- No organizational / human realities
- Errors assumed to be caused only from coding errors (not due to Microsoft!)
- Stable staff (no medical leaves, reassigned staff, etc)
- Other variables not captured in model (complexity and urgency of projects, new technologies)
Project Assessment

- Less skeptical to SD
- SD is an efficient tool to understand behavior
- To infer absolute results requires an extensive tuning process of the model
- Results seem to reflect well reality around current values
Thank you!