28TH ANNUAL
PACIFIC NW
SOFTWARE
QUALITY
CONFERENCE

OCTOBER 18TH – 19TH, 2010

ACHIEVING QUALITY IN A COMPLEX ENVIRONMENT

Conference Paper Excerpt from the

CONFERENCE PROCEEDINGS

Turning Complexity into Simplicity

Jon Bach

jonb@quardev.com

Twitter: @jbtestpilot

Abstract

This paper is an informal study in recognizing complexity and overcoming it to solve a testing problem. It enlisted the support of the Weekend Testers, a group of passionate professional testers in India who invite testers from all over the world to collaborate in weekend testing exercises. They asked me to be the host of one of their sessions, and this paper is about what happened when I gave them something complex to test.

Biography

At Quardev, Jon Bach is lead consultant and Manager for Corporate Intellect. He manages testing projects ranging from a few days to several months using Rapid Testing techniques like Session-Based Test Management – a way to manage and measure exploratory testing – which he and his brother James invented. In his 15-year career in testing and Quality Assistance, Jon has led projects for many corporations, including Microsoft, HP, Rational, and LexisNexis. He has presented keynotes and testing topics at over 60 testing conferences; written for testing magazines, and teaches testing techniques at Quardev and abroad.

Experience Report

Webster's definition of complexity seems like it has a sense of humor. They say it's "something complex" or "the quality or state of being complex."

That encourages one to look up "complex" which is more useful:

"Hard to separate, analyze or solve."

Sometimes complexity feels like trying to understand the cockpit of a 737. I talked to a pilot of such an airplane who, when asked how he deals with its complexity, told me: "I don't see it as complex. I only need a few of those switches and gauges at a time to do any one thing."

The secret seems to be that the pilot has a way of breaking down all of that stimulus, ignoring what he doesn't need and focusing on what he does need. While the definition of complexity says this is hard to do, I wanted to study how it could be made easy. (For example, a pilot's procedures and checklists are one way to battle the complexity of aircraft systems that need to work together to get people from one place to another.)

To study this juxtaposition further, I enlisted the Weekend Tester (India) crew to have them test what I thought was a complex application: Mifos, a microfinancing application by the Grameen Foundation. I chose this because I knew someone who worked for Grameen, who had asked me to give my impression of Mifos months ago.

I said it was one of the most complex things I had ever tried to model.

My charter was "Create a list of features toward creating a test coverage outline and for planning charters to be posted on the site for volunteer testers can run. Find every feature or enhancement you can find and be ready to discuss your questions and issues with project stakeholders."

I found over 250 testable features:

Feature Outline:

http://test.mifos.org:8085/mifos/AdminAction.do?method=load

- Photo
- Loan links
- Add a note
- See all notes
- Edit membership
- Office
  - Add
  - Define
  - View
  - Manage
  - Edit
- System users
  - View
  - Define new
  - Manage roles and permissions
- Perf history
  - Number of clients
  - Amount of last group loan if applicable
  - Average loan size for individual members
  - Total outstanding loan portfolio
  - Portfolio at risk
  - Total Savings
  - Loan Cycle number
  - Amount of last loan

- Total # of active loans
- Delinquent Portfolio

Schedule meeting

Roles
- Add new role
- Admin
- Loan Officer
- Manager
- Teller

Organizations
- View fees
- Define new fees
- View funds
- Define new fund
- View Checklists
- Define new checklist

Products rules and attributes
- View product categories
- Define new category
- View lateness/dormancy definition

Change Log

Apply Adjustment

Manage Loan Products
- View loan products
- Define new loan product

Manage Savings Products
- View savings products
- Define new savings product

Fees
- Fee calculation (mathematics / algorithms)
- Product fees
    - EF
    - ELDF
- Client fees
    - Application
    - Admin
    - Processing

View Office Hierarchy

Manage clients
- Create new group
- Create new member

Manage accounts
- Open new loan account
- Open new savings account
- Enter collection sheet data

Client accounts and Tasks
- Manage collection sheets
- Enter collection sheet data
- Print collection sheets

Create new clients
- Create new center
- Create new group
- Create new member

Create new accounts
- Create savings account
- Create loan account
- Create margin money account

Manage account status
- Change account status
    - Partial Application
    - Pending Approval
    - Active / Approved
    - Cancel
    - On Hold
    - Close
    - Blacklisted

Search

Your Settings

- Edit Info
- Change password

Login / Logout

Reports
- Client Detail
- Center
- Status
- Misc
- Performance
- Loan Product Detail
- Analysis
- Branch Office (dropdown list)
- Loan Officer (dropdown list)
- Loan Product (dropdown list)

Print Collection Sheets

Administrative tasks

Manage organization
- System users
    - View system users
    - Define new system user
    - Manage roles and permissions

Offices
- View offices
- Define a new office
- View office hierarchy

Organization Preferences
- View fees
- Define new fees
- View funds
- Define new fund
- View checklists
- Define new checklist
- View holidays
- Define new holidays
- Define accepted payment types
- View PPI settings
- Configure PPI Settings

Data display and rules
- Define Labels
- Define Lookup Options
- Define mandatory/hidden fields
- View Additional Fields
- Define additional fields

System Information
- View System Information

Manage products
- Product rules & attributes
- View product categories
- Define new category
- View lateness/dormancy definition

Manage product mix
- View products mix
- Define products mix

Manage Loan products
- View Loan products
- Define new Loan product

Manage Savings products
- View Savings products
- Define new Savings product

Manage accounts
- Manage Loan accounts
    - Reverse Loan disbursal
    - Redo Loan disbursal

Manage reports
- View admin documents
- Upload admin documents

View reports templates
Upload report template
View reports category
Define new report category
Manage surveys
View surveys
Define new survey
View question bank
Define questions

* went here to find new features:
http://www.mifos.org/developers/testing/1.1-test-strategy#objective-1

Adding and Removing Group Membership
Administrative Documents
Configuration
Group Loans with Individual Monitoring
Holiday Handling
Limiting Product Mix
Loan Defaults
Loan Schedule Independent of Meeting Schedule
Multiple Adjustments
Redo Loan Disbursal
Surveys
Undo Loan Disbursal
Help

Went to Roadmap to see other features:
http://wiki.java.net/bin/view/Javatools/RoadMap
BIRT reporting system
CGAP reports
PPI
Increased Support for Individual Clients/Teller Model:
- Ability to add group membership and to remove group membership to/from individual clients
- Ability to disburse loans on non-meeting days
- Ability to schedule repayments on non-meeting days
- Receipt and voucher printing from account pages
Accounting & adjustment enhancements:
- Undo loan disbursal / Redo loan disbursal
- Ability to adjust multiple loan adjustments
- Off-setting
Product Definition Flexibility:
- Loan defaults based on previous loan amount or loan cycle
- Calculating mandatory savings deposit amount on outstanding loans size
Moratorium Requirements
- Bulk loan creation
- Ability to restrict loan product mix
- French Localization
Enhancements
Reduce runtime of unit test suite to < 10 minutes
Improve handling of localized text
Upgrade Hibernate
Fix Look-up Value Overwriting
Support for additional lending models:
- Joint liability group (members of a group are held accountant able for loan repayments of others in their group; rules can vary, but examples include: group members can't receive a new loan a member of their group has a loan in arrears;)
- MGG
New Products:
- Insurance
- Shares
- Term Deposits
Increased Loan Product Flexibility:
- Additional repayment options: daily, flexible definition (ie, able to define outstanding amounts by month), able to edit specific repayment amounts
- Support for multiple (trache) disbursements for same loan, balloon repayments, etc
Enhanced Reporting Capability
Accounting Tools:
- Robust loan rescheduling

- Collection against write-offs

Adjustment Tools:
- Ability to adjust a single historical loan payment
- Ability to adjust single historical savings deposit/withdrawal/interest

Data Migration Tools:
- Tools for manual data entry of historical data
- XML (or another format) support for automated data migration

Accounting Interface: via batch file

Offline support: for Loan Officer daily tasks

Archive support: Ability to define rules for trimming database and summarizing data (ie, after N years, save only year end balances for savings accounts and archive savings transactions).

Branch level holidays

Support for additional lending models:
-SHG/Sacco support that tracks individual repayments separately from group repayments
- Full teller-model support: Removing meeting requirement in Mifos

Multicurrency Support?

Enhanced MFI Configuration Settings:
- Data scope configuration (can a branch see data from another branch)
- Additional levels of office hierarchy
- Defining lending models by branch

Enhanced Product Configuration Settings:
- Product availability by branch
- Products by lending model

Enhanced Loan Functionality & Flexibility:
- Automatic Calculation of Penalties
- Collateral Tracking
- Business Performance Tracking (can be handled via surveys?)
- Changing way loan cycles are handled
- Linkages to savings and/or Shares: Balance and Ownership requirements
- Configurable rules around early loan repayment
- Interest due calculation based on actual payments
- Payment via account transfers

Enhanced Savings Product Functionality:
- Savings account fees
- Savings acct restrictions (min balance, min amount to receive interest, max withdrawal amount, max # of withdrawals, etc)

Enhanced Client Data Collection:
- Biometrics collection
- Clients can belong to multiple groups/centers
- Codification of village/city/towns

Enhanced Work Flow & Permissioning System:
- Configurable Work flow management tools
- Field Level Permissioning
- Ability to combine loan and client/group approval steps into single step, ie, create client/group/center/loan all at same time

Interfaces:
- ATM integration
- Cash Management Systems
- Front-end POS devices
- Payment Systems
- Regulatory Agencies
- Kiosks
- Mix Market
- Smart Cards

Cash Management Support:
- Includes things like track clearing and tracking, deposit tracking, bank account balance tracking and management of multiple accounts, etc
- Tracking fund balances

Securitization Support:
- Portfolio tagging at account level
- Report generation

Misc:
- Easy tools for "Splitting" a Branch
- Ability to group clients into "Programs" (ie, HIV program, beggars program, etc)
- Set-up Wizard
- Localization into additional languages
- Self-extracting Windows Installer
- Collection of structured data in "Additional Data Fields"
- Support for variable/floating interest
- "Cash taken to Field" added to bulk-entry

The magnitude of this application stuck with me for months. I could take any one of those features and ask handfuls of questions about it, building thousands of tests. I could take one line in this outline like "Support for variable/floating interest" and have meeting after meeting on what it means and what it could mean. I could use 36 different test considerations in seeing how each line item feature interacted with one or many of the other features in that list. Then when testing, there'd be another layer of questions and impressions I would have, informing new tests and increasing my perceptions of what it did.

So when it came time to suggest a theme as host for Weekend Tester session #32, I knew it would be interesting to see what they thought of the complex Mifos application.

I gave them a charter:

"Read the new user manual – http://en.flossmanuals.net/bin/view/Mifos/WebHome. Also read the welcome (http://en.flossmanuals.net/bin/view/Mifos/Welcome) to learn a bit about the purpose of Mifos and pick one of the following sections to discover new bugs, usability improvements, and user manual improvements."

Testing wasn't all that important to me for this session, actually. I just wanted to discuss the notion of complexity and how it affected test planning.

But I didn't expect the conversation to be enhanced by the fact that we used a very SIMPLE application to collaborate our test notes. We used typewithme, a free online application that allows real-time notes to be displayed by whoever decides to type them in the window. It's a very simple application – nothing to set up, just a link to email anyone who you want to share the session. That's it. No layers of features, no permissions, no formatting to worry about, just open a new session, share the link, and type.

From that experience with simplicity and complexity back-to-back, 3 lessons about complexity emerged:

1) **Complexity involves many factors working at once**.

2) **Complexity involves being overwhelmed**.

3) **Complexity is an emotional relationship with the object of study**.

1) Complexity involves many factors working at once.

One tester said: *"I stopped reading the manual because there was too much to read (too complex). I felt I had to try the product to get context then re-read the manual. It was only at that stage that I could read the manual in detail to find issues with the wording, etc. I found the manual was non-technical, and I needed some technical content to understand how to enter data in the fields - eg. the limits of the fields, or field types, or basic date rules."*

Remedy:

Remember that this Weekend Tester session involved a live chat via typewithme, so it was easy to collaborate. It was in that public discussion where one tester suggested, "Get it to a base state and build from there."

This reminded me of the Stone Soup fable: a traveler comes to a village in search of food. Hearing that they have none, he asks for a pot and a fire to boil some water. He takes a rock and tells them it will magically make food for all to eat. He puts the rock in and invites all to taste it. "Boiling water?!?" says one, "I have a carrot. At least it will have some flavor." "Hot carrot water?!?" says another "I have a sweet potato I can add." And one by one, villagers begin to make a soup adding one vegetable at a time. By doing so, it becomes more complex than hot water. In this fable, the point **was** to add more layers (flavors).

There is a procedure to aid this kind of modeling. The "General Functionality and Stability Test Procedure" (GFS) is a testing and modeling process created by James Bach when consulting for Microsoft in 1999 to help them come up with a way for third-party software to earn the "Certified for Windows" stamp in 6 hours, no matter how complex the application. (For details, see http://www.satisfice.com/tools/procedure.pdf)

It involves doing a feature analysis (as I did above for Mifos) and categorizing features into two categories: "Primary" or "Contributing". A primary feature is one in which it is so important, that if it was broken it would make the product unfit for its purpose. A contributing feature either supports a primary one or stands alone, but if it were to have a primary flaw, the product would still be able to solve its primary goal.

2) Complexity involves being overwhelmed. During the Weekend Testing session with Mifos, a tester said: "The manual was too complex, but not the wording. It was written in a very basic way. It was just too much too soon. My brain was not prepared to read all that information. I had to try to learn it quickly in increments. That meant doing about 3 passes over the manual as I was learning the data entry form."

Remedy:

One way to fight complexity is to break up testing into time-boxed sessions. Instead of written test procedures or cases, you can spend time exploring a charter – a mission statement for a session. Each charter comes from a few minutes modeling the product (as I did above) and distilling test ideas into two or three sentences to guide the tester for the next hour or so. This limits the tester's attention on purpose. Working in session toward a mission is meant to focus and direct the tester – to simplify the complexity of the product so it's easier to test.

This is exactly why I wanted to start with a charter for the Weekend Testing session. I knew the testing was time-boxed and adding a time pressure to missions can make it seem complex. It can be overwhelming to meet a goal knowing that every minute must be used wisely.

Another tester said, "I find that learning in waves works best for me. I learn the structure of a problem/content first, then I learn the major components, and finally I fill in the gaps. I require my learning materials to be presented in a similar way."

The most common way to distill complexity happens every day on software projects -- get more eyes on the problem than just yours. Each person on the team takes a piece (a feature, for example) and makes it their own. Pairing, so one person isn't doing all the thinking and analyzing, is a possible technique to fight complexity. Have someone brief you, or make a bold boast and agree to train someone else on a feature you're confused about – that service mindset may inspire you to learn in a new way – to see it through new eyes.

3) **Complexity is an emotional relationship**: Whether it be confidence, safety, or familiarity, a tester told me, “You have to acknowledge the level of detail you are at, and move between your 'safe zone' and the complex zone. There is always a level you are confident with. If you freak out, then go back to your 'safe place' and start again.

Remedies:
That same tester said: “If we model at each level or state, then we improve our level of understanding, and the depth of confidence with complexity. It’s just like a baseball game, and you move from base to base, eventually scoring a run. Modelling helps us find a safe place when looking at complex pieces.”

I heard “safety” and “confidence” in that remark – two emotional considerations that might be necessary to help someone cope with data overload. Most of the testers in this session were not familiar with financial applications, but two of them were. Whether they were or not, whenever someone encountered an emotion of complexity, they seemed to let the group know, as if to set expectations correctly.

Diagrams or visual models are meant to intentionally limit the amount of information coming through. It’s a change in perspective. It’s as if you take a pen and paper and draw a simplification of the dials and gauges in an airline cockpit, attempting to describe its essential functions to someone who had never seen one. But upon seeing the model, they may immediately abstract it to something that’s similar to what they *do* know. They may even have an epiphany of some kind of analogy, like “Testing is like a bird looking for worms.” That kind of attempt to find meaning and familiarity turns complexity into simplicity.

With typewithme, people seemed not only delighted by its simplicity. They had mastered it in seconds because it was a familiar paradigm and did not restrict or force them to think in a way to which they were unaccustomed. In fact, many wanted to know **more** of its other features to the degree that when asked to do another Weekend Tester session, I made exploring its features the primary mission.

Conclusion

The experience with group testing Mifos showed me that though complexity is a perception, it is also a force of many factors acting at once, overwhelming you, causing an uncomfortable emotion which may promote a desire for safety and familiarity.

It’s easy to say “just take it one step at a time” in fighting complexity, but even to say that much means being aware of what complexity is for different people. Are they an airplane pilot, unafraid of all the dials and switches in front of them because they understand the value and purpose of each? Or could it be that under pressure with inadequate emergency training, that same pilot would become paralyzed because of emotional overload?

The experience of this overseas collaboration in real-time with a two real products demonstrated validated my first experiences with Mifos as a complex application, but it also gave me some insight by way of the three lessons above as a starting point to understand when and how to confront (simplify) complexity.