Testing Calculation Engines using Input Space Partitioning & Automation

Thesis for the MS in Software Engineering

Chandra M. Alluri

calluri@gmu.edu, chandra_alluri@freddiemac.com

The Volgenau School of Information Technology & Engineering
Department of Computer Sciences
George Mason University
Fairfax, VA USA

Introduction

- Financial services applications contain several subsystems that involve complex calculations
- In a particular application, multiple calculations needed to be performed by different calculators to achieve the business objective. These calculators together can be termed as the ***calculation engine***.
  - e.g. Financial models, Valuation models, Pricing models, Risk assessment models
- Errors in calculation engines inflicts severe damages
- Companies, internal and external auditors rely on functional testing results to determine the quality of the systems
- Test approach extends Category Partition Method
- Approach is validated by several case studies

Presentation Outline

- Challenges in testing
- Test approach (proposed)
- Test case design - Input Space Partitioning
- Test case design - Modeling with Tool
- Case studies
- Results
- Pros and cons
- Framework (recommended)
- Conclusion

Motivation

- Unfamiliarity with the software
- No user interface
- All computations occur on the server
- Specification factors
- What confirms the quality?
- When to stop the testing?
- Testers familiar with business but not testing and vice-versa
- How the testing can be automated?

- How can this be consistently tested?
- What are the similarities?
- How to manage large sets of data?
- Inconsistency in writing the requirements
- Audit relies on the system testing results
- Conventional automation does not work

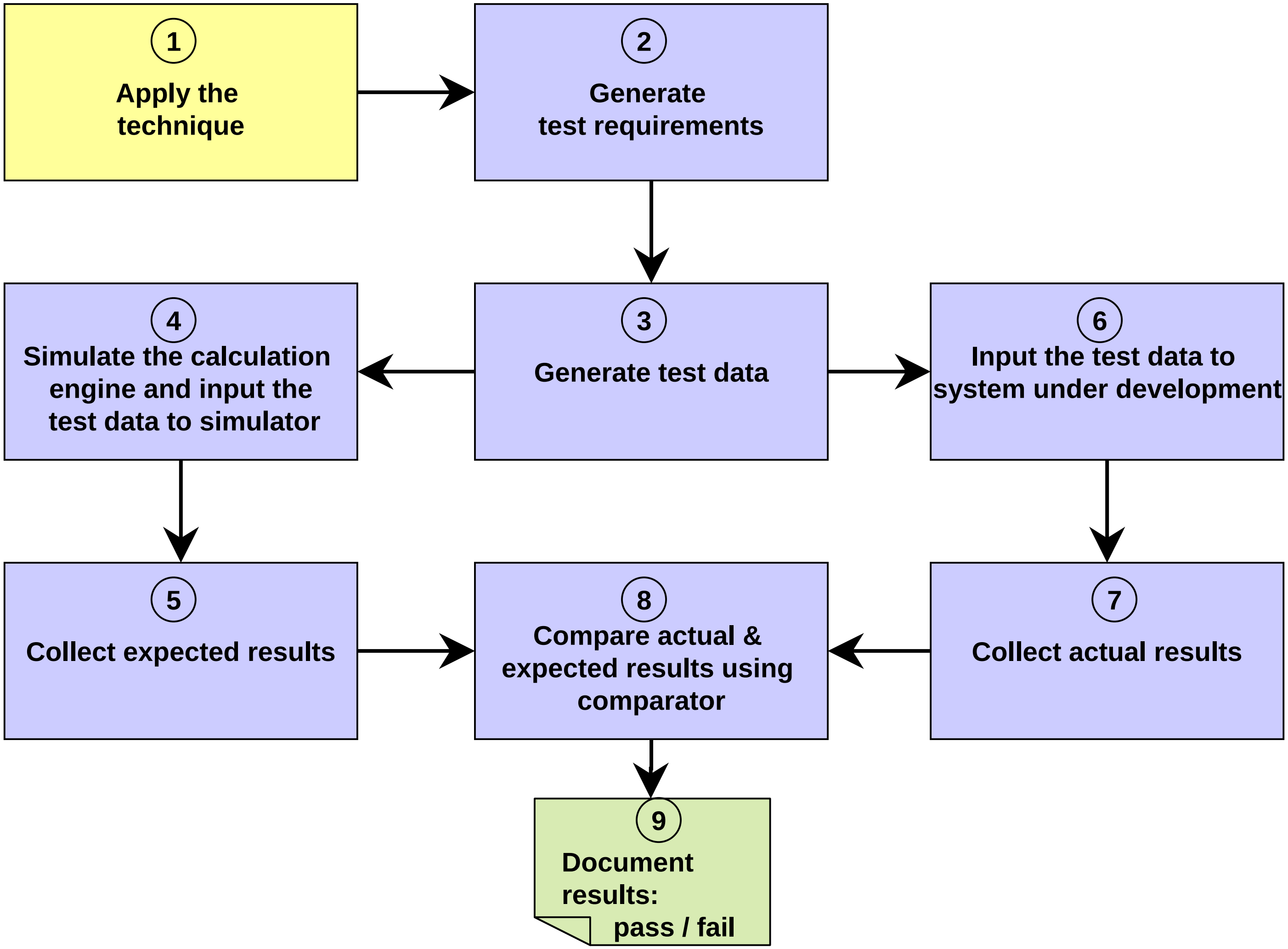
Characteristics of Calculation Engines

- Low controllability
- Low observability
- Specification factors
  - Precision, truncation, and rounding
- Design or implementation characteristics
  - Pricing grids
  - Data flow
  - Conditional events
  - Calculation algorithms
  - Architecture
  - Important attributes
  - Intermediate values
  - Business cycles

Test Approach

- Traditional approach applies black-box testing techniques
- Testing calculation engines involves testing combinations of the inputs
- Test approach should improve controllability and observability
- A new set of automation tools are required
- Test data should be reusable
- Test process should be repeatable
- Test approach should accommodate organizational strengths and weaknesses

Test Approach - Overview



Test Approach - ISP

- Input Space Partitioning (ISP)
  - Input domain is viewed as partitions and blocks for each or a group of characteristics

- Blocks and values
  - ✓ Valid values
  - ✓ Sub-partition
  - ✓ Boundaries
  - ✓ Invalid values
  - ✓ Missing partitions
  - ✓ Overlapping partitions

- Input combinations
  - ✓ Apply coverage criteria

Freddie Mac
We make home possible℠

Test Approach – ISP (cont.)

GEORGE MASON UNIVERSITY

Category partition method framework

1. Identify testable function
2. Identify implicit & explicit variables that affects behavior
3. Identify important categories or characteristics
4. Choose a partition or a set of blocks for each characteristic
5. Apply coverage criteria

- Testable Function
  - A small independent unit of requirements
  - Requirements set
  - Ex: Retrieve the market data from grids
  - Obtain the prices from PMA
  - Obtain the interest rates
  - Calculate the new prices

- Coverage Criteria
  - ✓ Base Choice (BC)
  - ✓ Multiple Base Choice (MBC)
  - ✓ Pair-Wise (PW)

Test Approach - Modeling

- Fusion Test Modeler (FTM)
  - Existing modeling tools do not address the testing needs
  - I developed FTM to address the Freddie's testing requirements
  - FTM facilitates modeling different specifications
  - FTM uses tree structure in modeling
  - Models are stored as XML files
  - FTM helps trace the test cases to the requirements

Freddie Mac
We make home possible℠

Test Approach – Modeling (cont.)

George Mason University

Identify testable function → Model the requirements using the tool → Apply coverage criteria

Logical expressions,
if-else structures,
use cases, and other
conditional requirements
can be modeled using the
FTM tool

FTM parses the graphs and generates the test cases

- Coverage Criterion
  - ✓ Prime path coverage

Case Studies

- Test approach is validated with the help of 4 case studies
- Freddie Mac Operations and Technology has 3 business divisions: Sourcing and Servicing, Investments and Capital Markets (I&CM), and Finance and Accounting
- Case study # 1 and 2 belongs to Sourcing business.
- The functionality in Case study # 3 belongs to I&CM
- The functionality in Case study # 4 belongs to F&A
- Modeling is applied only to case studies # 1 and 2
- ISP is applied to all the 4 case studies

Case Study # 1: Contract Pricing

- This case study encompasses 2 use cases
- Allows Freddie Mac to create, import, and price the contracts
- Applied in 2 stages

<u>First stage</u>

- Contract has 29 attributes
- BC is applied to all the attributes
- All the 200 requirements are satisfied by the BC and PW criteria

<u>Second stage</u>

- 7 Pricing attributes are isolated
- ISP is applied with BC, MBC, and PW.
- Requirements are modeled with the FTM

Freddie Mac

We make home possible℠

Case Study # 2: Loan Pricing

George Mason University

- This case study relates to one use case in Pricing sub-system of the Selling System.
- Any changes to the loan triggers the calculations in this use case
- This case study involves 3 entities: Loan with 140, Contract with 29, and Master Commitment with 50 attributes
- Among all these attributes, only 12 are involved in calculations
- Among the 12, 6 have constraints and the rest will take the values from grids
- ISP is applied with BC, MBC, and PW criteria
- Requirements are modeled using FTM

Case Study # 3: Amortization

- This case study involves the amortization of the Multifamily loans
- This has 16 calculators; 6 are preliminary calculations and the remaining 10 calculations occur recursively in a sequence, each feeding the output to the next calculator
- Testable functions are considered at the calculator level
- Important characteristics of the loop are identified; loop is treated as one of the partition for the testable functions
- ISP is applied with the BC criteria
  - MBC do not get additional coverage
  - PW produces too many invalid combinations
- Modeling is not applied

Freddie Mac
We make home possible℠

Case Study # 4: Calculate SEY IRR

GEORGE MASON UNIVERSITY

- This involves calculating SEY-IRR on certain conditions
- This case study is documented following the steps in the recommended framework of this thesis
- Case study documentation helps
    - How to first test and eliminate the invalid values
    - How to test with the valid values
    - How to treat the loop conditions
    - How to provide abstract values
    - When to provide real values
- ISP is applied with the BC criterion
- MBC and PW are not suitable for the same reasons explained in case study # 3
- Modeling is applied with a different scope – Modeling results are not considered for this case study

Results - Methods

- The effectiveness of the test approach on the case studies is measured by using the following tools and techniques.

  - ❑ Functional coverage – RTMs
  - ❑ Statement coverage – jTest tool
  - ❑ Logical correctness – Verifying the results of simulator and the SUT
  - ❑ Defect analysis – Defects are analyzed to check if the proposed approach could have prevented.
  - ❑ Improvements in the process
  - ❑ Consistency in the practice

Freddie Mac
We make home possible℠

Results – Functional coverage

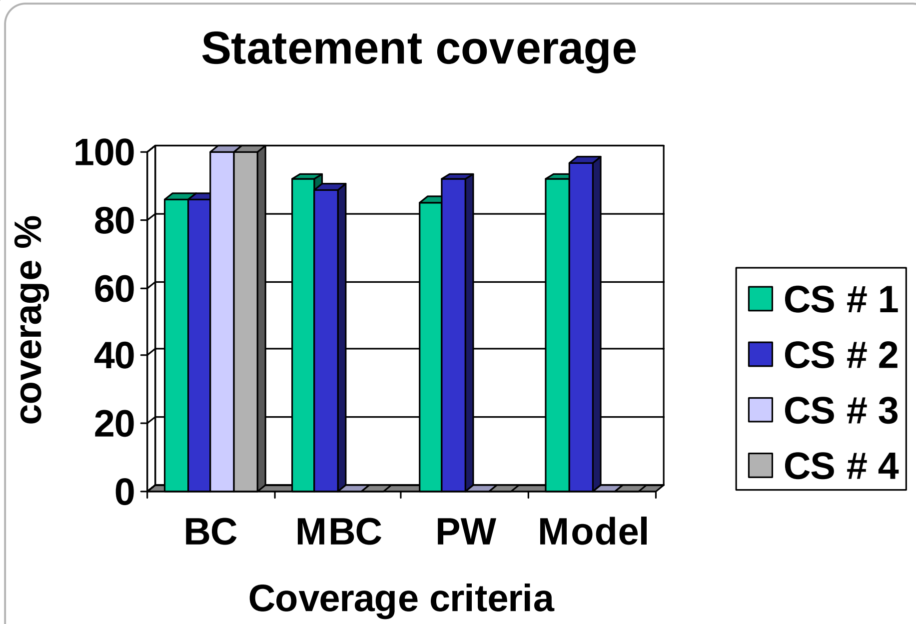
George Mason University

Case Study # 1					
No. of Tests	**Base Choice**	**Multiple Base Choice**	**Pair-Wise**	**Pair-Wise refined**	**Modeling**
Number of Tests	15	30	23	23	27
Functional Coverage		**100%**		N/A	**100%**
Case study # 2					
Number of Tests	26	52	72	N/A	131
Functional Coverage		**100%**		N/A	**100%**
Case study # 3					
Number of Tests	74	N/A	N/A	N/A	N/A
Functional Coverage	**100%**	N/A	N/A	N/A	N/A
Case study # 4					
Number of Tests	56	N/A	N/A	N/A	N/A
Functional Coverage	**100%**	N/A	N/A	N/A	N/A

Results – Structural coverage (1)

Case Study # 1					
No. of Tests	Base Choice	Multiple Base Choice	Pair-Wise	Pair-Wise refined	Modeling
Number of Tests	15	30	23	23	27
Statement Coverage	86	92	85	92	92
Case study # 2					
Number of Tests	26	52	72	N/A	131
Statement Coverage	86	89	92	N/A	97
Case study # 3					
Number of Tests	74	N/A	N/A	N/A	N/A
Statement Coverage	100	0	0	N/A	0
Case study # 4					
Number of Tests	56	N/A	N/A	N/A	N/A
Statement Coverage	100	0	0	N/A	0

Results – Structural Coverage (2)



- BC applied to all case studies
- MBC, PW, and Modeling are applied only to case study # 1 and 2.
- BC achieved 100% functional and structural coverage

Results (cont.)

- Logical correctness of the calculations is verified by comparing the outputs of the simulator and SUT
- Defect analysis
  - ✓ Defects of the Selling System are analyzed for the previous 8 releases. This approach if applied in the past would have prevented 75% of the defects
- Process Improvements
  - ✓ Only one use case was documented in the entire application as case study # 3. The entire application has nearly 600 test cases
  - ✓ Tests were derived early in the life cycle. Developers unit tested using these tests. Later system passed clean with 17000 records
  - ✓ Testing cycle was reduced to 0.5 days from 5 man days
  - ✓ This system went into production with 0 non-conformances
  - ✓ Very stable in production for one year with hardly any functional defects
  - ✓ Consistency in the practice

Pros and Cons - Modeling

- Modeling - pros
  - ✓ Instant generation of test cases from model
  - ✓ Easy mapping of test cases to requirements
  - ✓ Easy to identify critical paths in model
  - ✓ Common understanding of the model
- Modeling – cons
  - Skill set
  - Domain knowledge
  - Inconsistent modeling
  - Inconsistency in practice
  - Additional maintenance of the XML files

Pros and Cons - ISP

- ISP - pros
  - ✓ Easy to adapt
  - ✓ Test cases using coverage criteria BC, MBC, & PW are automated
  - ✓ Test cases are generated early in life cycles with abstract values
  - ✓ Easy trace to requirements
  - ✓ Repeatable and reusable
  - ✓ Offers variety in data
  - ✓ Increased controllability
  - ✓ Increased observability
  - ✓ Eases the data aging
  - ✓ Easy to cope with changes in the requirements

- ISP – pros (cont.)
  - ✓ Sub-modeling helps in testing the combinations
  - ✓ BC criteria simplifies business rules testing
  - ✓ Business cycles testing made easy with ISP

- ISP – cons
  - ▪ Depends on the testable functions
  - ▪ ISP is efficient with automation
  - ▪ Coverage criteria should be chosen carefully

Framework

Mapping of the steps in Framework to Freddie Mac's SDLC methodology

Initiation	Requirements Analysis	Design	Construction	Acceptance	Implementation
	Step # 1 Identify the scope	Step # 5 Apply BC Criteria with invalid values	Step # 11 Prefix the test cases with other values		
	Step # 2 Identify the testable functions	Step # 6 Eliminate invalid values from each partition	Step # 12 Provide real values		
	Step # 3 Identify the implicit & explicit attributes	Step # 7 Apply BC, MBC, PW appropriately with valid values	Step # 13 Build the simulator, input test data and derive expected results		
	Step # 4 Identify distinct valid and invalid abstract values for the attributes	Step # 8 Identify constraints in the testable function	Step # 14 Input the test data to SUT and derive actual results		
		Step # 9 Exclude invalid test cases from the suite	Step # 15 Compare actual & expected results		
		Step # 10 Ensure the functional Coverage with RTM			
	Test Strategy	Test Plan Test Data Management Plan Test Cases RTM	Test Data Test Results Execution logs Test Summary Report Final RTM		

..... Activities in dotted lines are iterative

SUT: System-Under-Test

RTM: Requirements Traceability Matrix

Conclusion

- Results show ISP achieves high functional and structural coverage consistently
- The framework is adaptable to the software development process, independent of technology and development methodology
- Test effectiveness depends on choosing the testable functions
- Sub-modeling is preferred over pair-wise coverage criteria
- Although case studies are performed at the system testing level, this can be extended to the unit and user acceptance testing
- The case studies chosen are both universal and very critical for any financial services applications – This framework is simple to understand and easy to manage. And can be extended to other similar applications.
  - ✓ 2 new projects are adopting this approach

Conclusion (cont.)

- Agility
  - ✓ Agility to change test cases when requirements change
  - ✓ Introduction of the new products
- Respond to the changes
  - ✓ Identification of the testable functions helps in impact analysis of the changes
- Estimations
  - ✓ A brief understanding of number of tests with abstract values helps in estimating the efforts

How the challenges are addressed?

- Test approach addressed the following challenges
  - ✓ Common characteristics of the calculation engines are documented
  - ✓ Critical testable functions and test cases are identified
  - ✓ Controllability and observability are unraveled
  - ✓ Risk of not knowing the internals is mitigated
  - ✓ All the events that triggers calculations absorbed by the ISP tests
  - ✓ Consistency is achieved
  - ✓ High functional and structural coverage assures the quality

Further Work

- Automatic generation of test data
- Automatic filtering of invalid combinations
- Building the user interface
- Automatic coverage detection
- Testable functions – estimation technique