CONCORDIA UNIVERSITY &

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE – MONTRÉAL - CANADA

Scenario-Based Black-Box Testing in COSMIC-FFP

Manar Abu Talib, Olga Ormandjieva, Alain Abran, Luigi Buglione

2nd Software Measurement European Forum 2005

Agenda

- Introduction
- COSMIC-FFP Measurement Method
- Scenario-Based Testing
- Using Entropy for assigning priorities to test cases
- Discussion and Next Steps

Introduction

- *At SMEF 2004, papers were presented on :*
  - *Functional sizing with COSMIC-FFP*
  - *Functional Complexity measurement with Entropy concepts.*
  - *Both types of measurement methods share significant similarities & some differences*
    - *Presented at* ***IWSM 2004*** *in Berlin*

Introduction

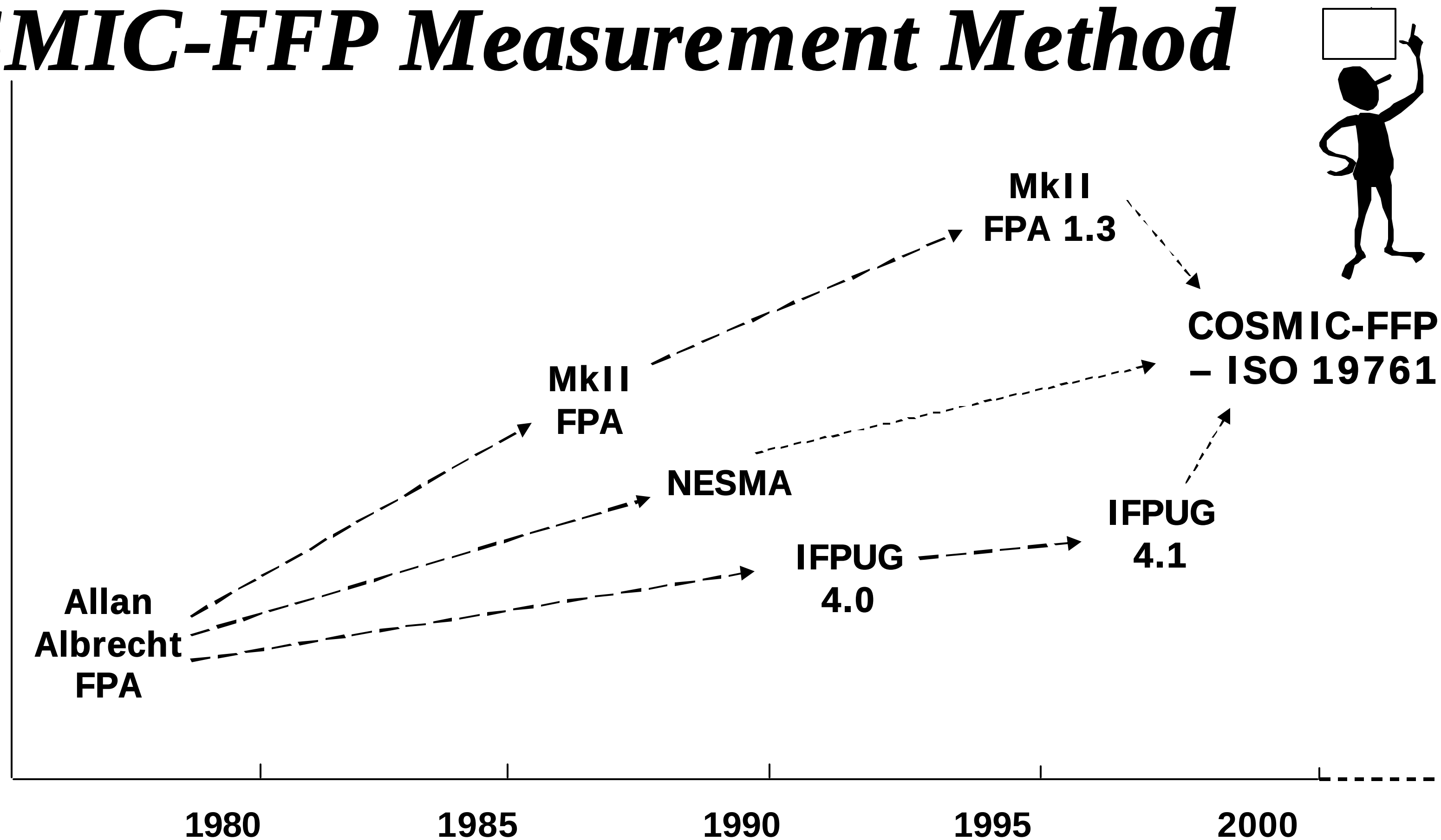
- *Functional complexity and Entropy are playing a significant role in testing – in general*
- *Introducing Entropy in software testing strategies could be of interest.*
- *In this presentation we investigate a contribution of COSMIC-FFP in introducing testing strategies and in leveraging the benefits of using Entropy concepts to assign priorities to test cases.*

Context 1

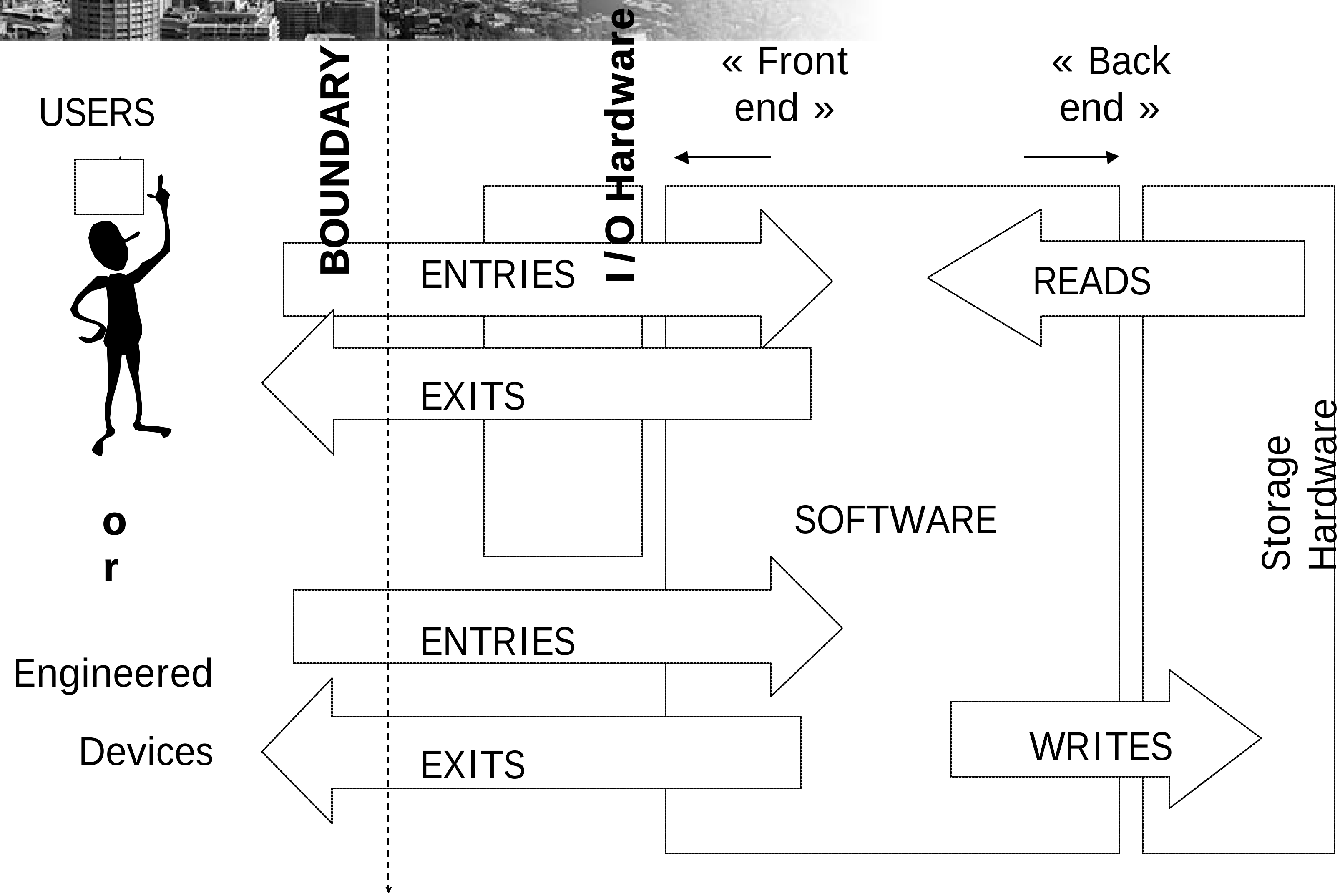
☞ ***Software Engineering****: A* ***discipline*** *for the systematic production and maintenance of large and complex software systems*

☞ ***Software Measurement: a mechanism to provide feedback on software quality***

COSMIC-FFP Measurement Method



COSMIC-FFP Measurement Method

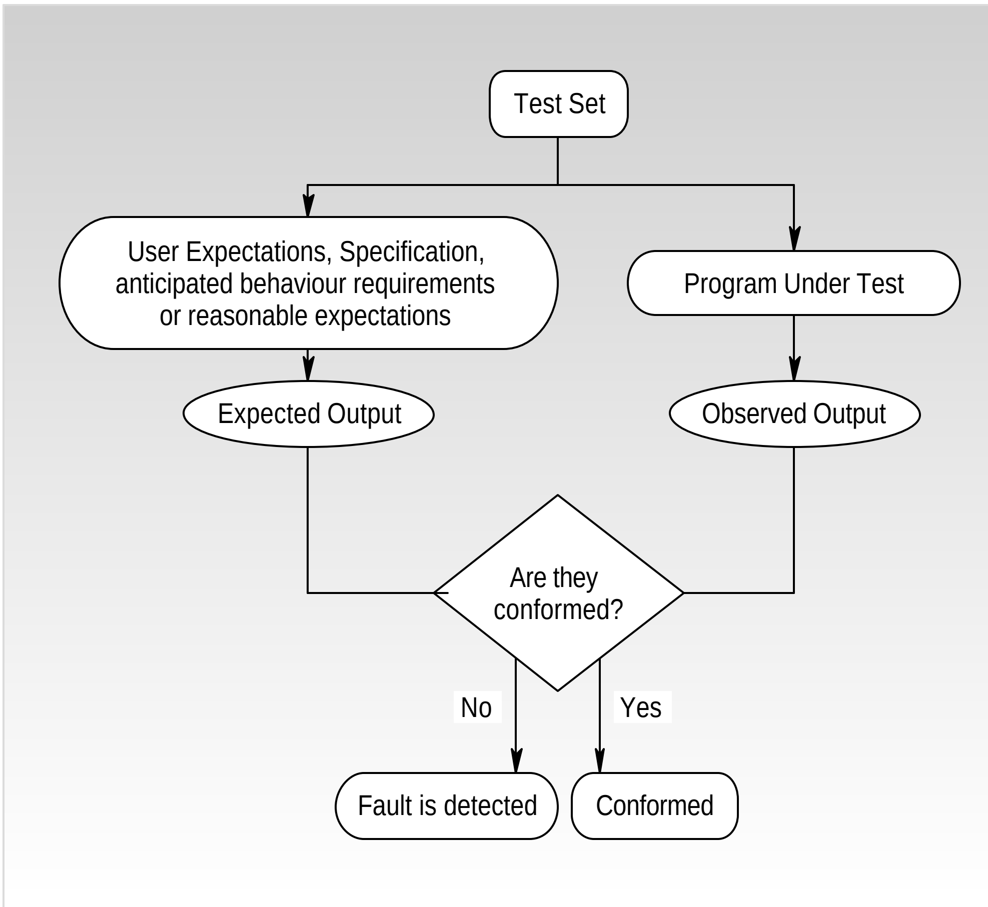


Context 2

☞ **Testing:** *Software testing consists of the dynamic verification of the behaviour of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the specified expected behaviour*

(SWEBOK definition – ISO TR 19759)

« Expected » means:



Oracle Methodology

Context 2

Types of Testing:

Black-box:

- test cases are generated and executed from the specification of the required functionality at defined interfaces

White-box:

- test suite is generated from the implemented structures

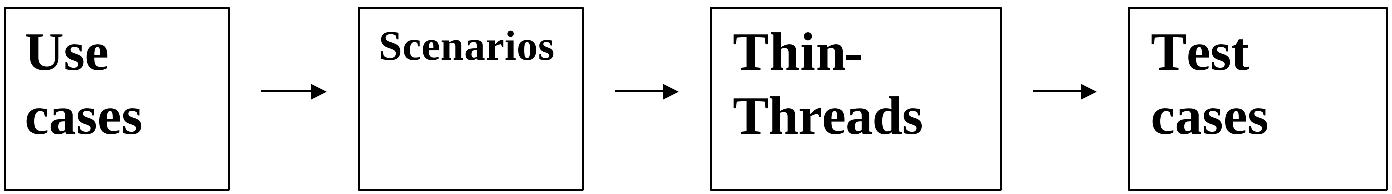
Gray-box:

- the modular structure of the implementation is known but not the details of the programs within each component

Scenario-Based Testing

- *Scenario based testing is a typical black box testing methodology at the system level*
- *Scenarios depict the sequence of executions of the system, and the test cases can be derived from the use-case model and its corresponding UML diagrams*

Related Work on Scenario-Based Testing



Use Case: is defined as a collection of related scenarios that describes actors and operations in a system

Scenario: a specific sequence of actions and interactions between actors and the system

Thin Thread: is a minimum usage scenario in a software system. The execution of a thin thread demonstrates that a method performs a specified function

COSMIC-FFP and Scenarios

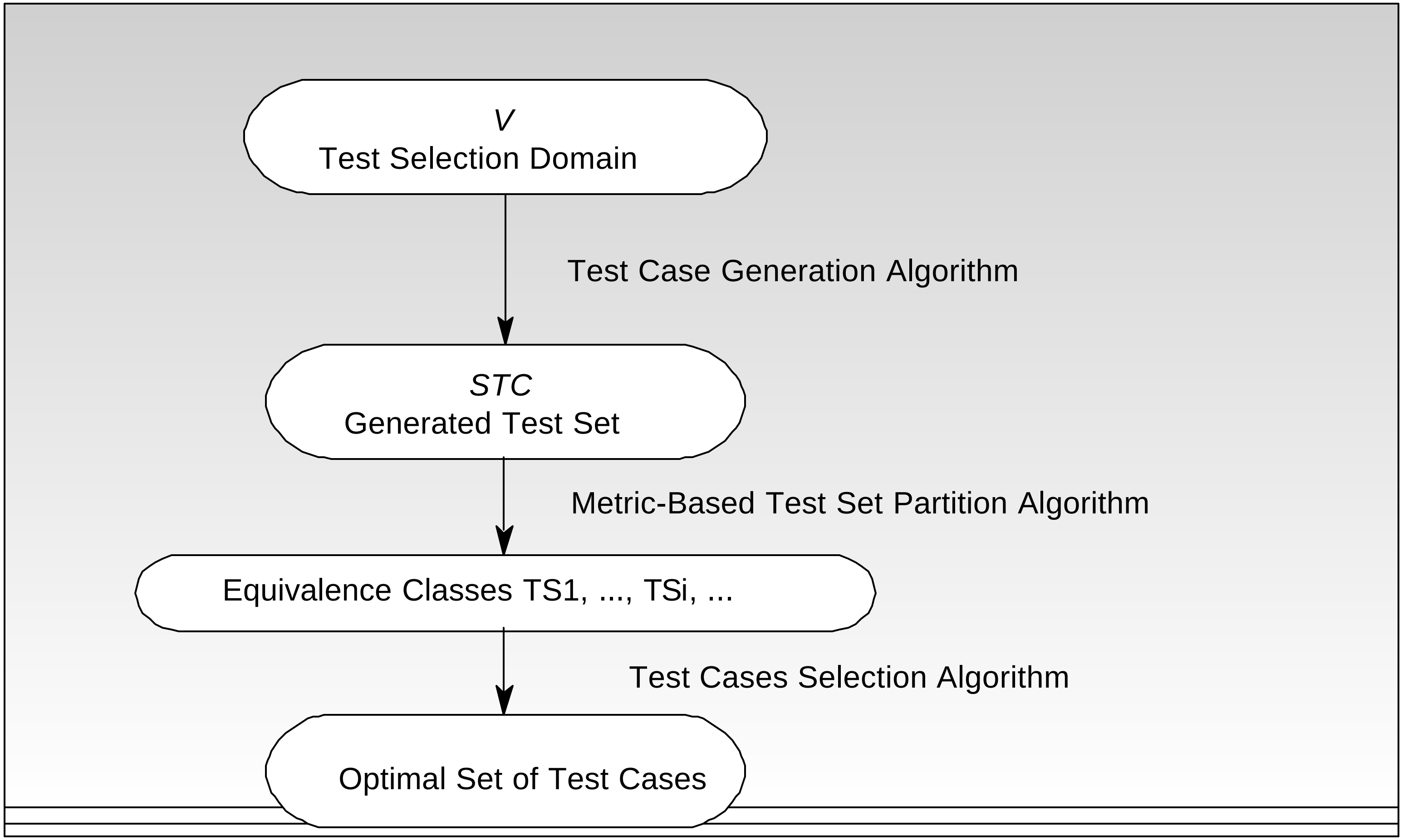
COSMIC-FFP Concepts	Related UML Concepts
Boundary	Use case diagram
User	Actor
Functional process	Use case
Data movement	Operation (message)
Data group	Class
Data attribute	Class attribute

COSMIC-FFP Concepts and their related UML concepts

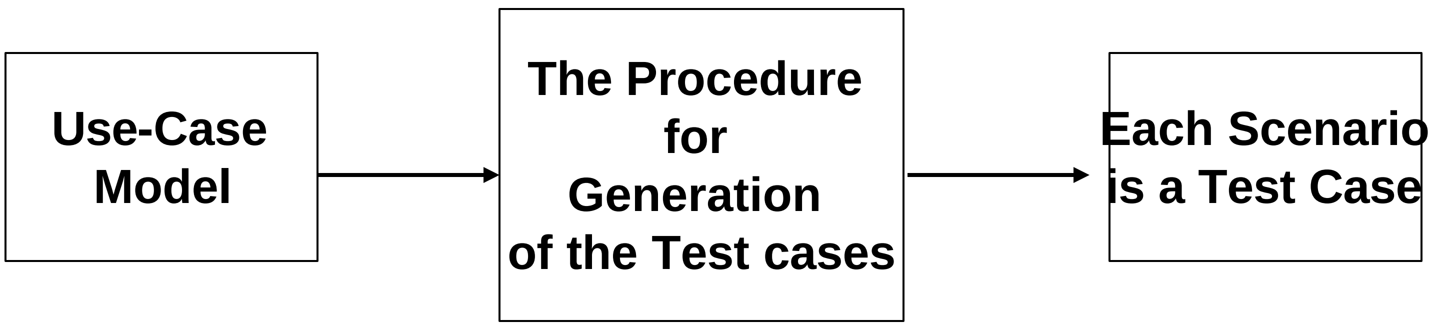
Scenario-Based Testing with COSMIC-FFP

- *Goals:*
  - *Reduce the number of test cases while keeping the highest fault coverage within the budgetary constraints*
    - ***Fault coverage:*** *The power of a test cases generation technique to detect faults in an implementation*

Our Testing Approach



Test Case Generation



- *A test case is mapping the scenarios to a sequence of events in time (or data movements in COSMIC-FFP)*

Metric-Based Test Case Partitioning Algorithm

Precondition :{ $STC \neq \emptyset \wedge \varepsilon max > 0 \wedge$ " $i \cdot TS_i = \emptyset$}

-sets εmax (predefined value based on experimental work)

While $STC \neq \emptyset${

- selects the longest test case *t* in set *STC*

-sets ε to 0

- moves *t* from the set *STC* to the set *TSi*

While ($STC \neq \emptyset \wedge$ ($\varepsilon < \varepsilon max$) {

-chooses a test case *t* in set *STC* whose distance to the set *TSi* is minimum in order to put similar test cases into one equivalence class

-sets ε to *the* distance between t and TS_i

-if $\varepsilon < \varepsilon max$ then moves *t* from *STC* to set TS_i

} // equivalence class TS_i *is created*

i=i+1;}

Distance between Test Cases

- ***Distance between two test cases t1 and t2 is:***

*td (t1, t2) = similarity (t1 , t2) * dissimilarity (t1 , t2) * e ^ l*

Where:

- ***l*** *= -(length (t1)/length (ts))*
- ***The similarity (t1 , t2)*** *= 2 (- length (LCP (t1, t2)) where LCP is the longest common prefix of the two test cases. The range of the similarity measure is between 0 and 1*
- ***The dissimilarity*** *measure between two test cases t1 and t2 is calculated as the number of elementary transformations that are minimally needed to transform the string (t1/LCP (t1, t2)) into the string (t2/ (LCP (t1, t2))*

Distance between Test Cases (Example)

- *The distance between the following two test cases: e1.e2.e3.e4.e1 and e1.e1.e1.e2.e1 = ½ * 3 * e ^ -1*

- *The distance formula td (t1, t2) indicates that the greater the distance you have between two test cases the more they will differ from each other*

Priority of Test Cases

- In our approach, the Entropy-based functional complexity measure (FC) is used to prioritize the test cases.

- The formula that is used to calculate the functional complexity is as follows:

$$FC = -\sum_{i=1}^{n} (f_i / NE) \log_2 (f_i / NE)$$

- More and diverse functionality of the system would lead to bigger portion of the system involved in that usage.

- The entropy calculated on a sequence of events abstracting a scenario quantifies the average information interchange for a given usage of the system.

Functional Complexity (FC) of a Test Case

- *FC is calculated on a sequence of events: abstracting a scenario <u>quantifies</u> the average information interchange for a given usage of the system*
  - *Example: FC of sequence (e1.e2.e3.e4.e1) for e1 has a higher average information content than sequence (e1.e1.e1.e2.e1)*
  - *It is then more complex than another and it will have greater priority than the second test case*
    - *Eg. more system usage*

Test Selection Algorithm

For all non-empty equivalent classes *TSi*

Step 1. *Choose the highest priority test case from the equivalence class TSi;*

Step 2. *Add the chosen test case to the Optimal Set and remove it from the equivalence class TSi;*

Step 3. *Increase the total testing cost in C*

If (The total testing cost exceeds a given budget Cmax)

End the algorithm

End For

Conclusion

- *COSMIC-FFP method was extended for testing purposes by combining the functions measured by the COSMIC-FFP measurement procedure with the black box testing strategy*

- *Applicability of using Entropy measurement with COSMIC-FFP for assigning priorities to test cases*

Future Work Directions

- **Apply this testing strategy into case studies**
  - Will facilitate implementation in industry

- **Entropy is used in Reliability Estimation:**
  - **Investigate the use of COSMIC-FFP for the ‘reliability estimation’ of the software**

References

1. [1] ABRAN, A., DESHARNAIS, J.-M., OLIGNY, S., ST-PIERRE, D. AND SYMONS, C., COSMIC Implementation guide to ISO 19761, École de technologie supérieure - Université du Québec, Montréal. 2003, URL: http://www.cosmicon.com
2. [2] ABRAN, A., ORMANDJIEVA, O. AND TALIB, M.A. Functional Size and Information Theory-Based Functional Complexity Measures: Exploratory study of related concepts using COSMIC-FFP measurement method as a case study, 14th International Workshop of Software Measurement (IWSM-MetriKon 2004), Springer-Verlag, Konigs Wusterhausen, Germany, 2004, pp. 457-471.
3. [3] ALAGAR, V.S., CHEN, M., ORMANDJIEVA, O. AND ZHENG, M., Automated Generation of Test Suits from Formal Specifications of Real Time Reactive Systems. Submitted to a Journal, 2004.
4. [4] BAI, X., PENG, L.C. AND LI, H., An Approach to Generate Thin Threads from UML Diagrams. Technical Report, TR-03-12, Software Engineering Research Group, School of Computer and Information Science, Edit Cowan University, 2002.
5. [5] BAI, X., TSAI, W.T., FENG, K. AND L.YU, Scenario-based Modeling and Its Applications to Object Oriented Analysis Design and Testing, Proceedings of IEEE Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2002), San Diego (USA), 7-9 January 2002, pp. 253-270
6. [6] BEIZER, B. Software Testing Techniques. Van Nostrand Reinhold, 2/e, 1990, ISBN 1850328803
7. [7] BERTOLINO, A. Knowledge Area Description of Software Testing Guide to the SWEBOK. URL: http://www.swebok.org , 2004.
8. [8] CHOW, T.S. Testing Software Design Modeled by Finite State Machines, IEEE Transactions on Software Engineering, Vol. SE-4, No.3, 1978, pp.178-187
9. [9] DAVIS J.S. & LEBLANC R.J., A Study of the Applicability of Complexity Measures, IEEE Transactions on Software Engineering, Vol. 14 No.9, September 1988, pp.1366-1372
10. [10] EN-NOUAARY, A., DSSOULI, R. & KHENDEK, F., Timed Wp-Method: Testing Real Time Systems, . IEEE Transactions on Software Engineering, Vol. 28 No.11, November 2002, pp.1023-1038
11. [11] ISO/IEC19761. Software Engineering - COSMIC-FFP - A functional size measurement method, International Organization for Standardization - ISO, Geneva. 2003.
12. [12] JENNER, M., Automation of Counting of Functional Size Using COSMIC-FFP in UML, in Proceedings of the 12th International Workshop on Software Measurement (IWSM 2002), Magdeburg (Germany), October 2002, pp.43-51.
13. [13] LARMAN, C. Applying UML and Patterns, An introduction to Object Oriented Analysis and Design and the Unified Process. Prentice-Hall, 2001, ISBN 0130925691
14. [14] WEISS, S.N. AND WEYUKER, E.J., An Extended Domain-Bases Model of Software Reliability, IEEE Transaction on Software Engineering, Vol.14 No.10, October 1988, pp.1512-1524
15. [15] PHILIPPE KRUCHTEN, The Rational Unified Process : An Introduction, 2nd Edition, Addison-Wesley Pub Co, 2000, ISBN: 020170710

Thank You !

Questions?