

Infosys®

Win in the flat world

Model Based Software Test Plan Automation

Ravi Gorthi, Ph D
Test Automation Research Lab
Software Engineering and Technology Labs
Infosys Technologies Limited, Bangalore, India
email: ravi_gorthi@infosys.com

Topics

- An overview on Test Case and Regression Test Case Creation and Maintenance
 - Test Plan vs Test Execution
 - Creation and maintenance of test scenarios, cases, data and scripts
 - What are regression test cases? Creation and maintenance of regression test cases

- The relevance of this area to the Total-Cost-of-Ownership (TCO) of medium to large s/w applications
 - Typical number of test scenarios and cases in a medium / large s/w application
 - Efforts and costs involved in the creation and maintenance of test cases and regression test cases

Topics

- The state-of-the-art research and practices of this field
 - Model based (UML Activity Diagram, State Diagram based) manual and automated creation and maintenance of test and regression test cases
- Some interesting work done by the speaker and his team in this area for the last three years
 - The concept of 'Unit of Behavior'
 - Techniques for automatic generation of test cases and regression test cases from UML Activity Diagrams
 - Results of field trials of these techniques

Q& A

Background

- Software Testing
 - Functional and non-functional testing
 - Planning phase and Execution phase
- Planning phase of functional testing: creation and maintenance of,
 - Test scenarios: correspond to UML use-cases
 - Test cases: correspond to 'input, expected output' patterns
 - Test data: correspond to actual input and expected output values
 - Test scripts: correspond to code that the execution phase teams can use
- Execution phase of functional testing:
 - Creation and maintenance of required test execution environments
 - Execution of required test scenarios (include test cases, data and scripts)

What are Test Scenarios, Cases and Data?

- Test Scenarios

- Testing needs of a logical unit of SRS (e.g. use-case / a logical path in a use-case)

- Test Cases

- Test requirement descriptions related to each unit of stimuli-response
- Typically described in terms of <Inputs to software system, A set of conditions, Outputs from software system>
 - E.g. <Input: user-id and password; Condition: incorrect password; Output: error message-1>

- Test Data (corresponding to each test case)

- Specifications of test inputs, conditions and expected output
 - E.g. <Inputs: ravi_gorthi and abc2\$password; Condition: abc2\$password is an incorrect password; Output: “Invalid Password - enter password again”>

Background

- **Regression Testing**
 - Arises out of changes made to a software application
 - Functional and non-functional testing
 - Planning phase and Execution phase

- **Planning phase of regression testing: creation and maintenance of,**
 - Find changes (additions, modifications and deletions) to functional test scenarios, cases, data and scripts
 - Effect the required changes

- **Execution phase of functional testing:**
 - Find out the changes required to the test execution environments
 - Execution of required test scenarios (include test cases, data and scripts)

Relevance of Test Planning

- NIST report 2002
 - Software Errors Cost U.S. Economy \$59.5 Billion Annually
 - More than a third of these costs (\$22.2 b) could be eliminated by an improved testing infrastructure
 - An undetected defect, post software deployment, costs \$14K to fix
- Medium to large s/w applications are observed to have anywhere between 5K to 100K test cases
- Typically consumes 30 to 50% of *total testing efforts* (testing typically consumes 40 to 60% of total software TCO)
- Every 10% improvement in productivity and quality of testing can lead to saving of millions of dollars

Current state of affairs

- **Planning phase of functional testing:**
 - Functional and regression test planning is manual

- **Execution phase of functional and regression testing:**
 - Tools exist that aid creation, maintenance and execution of test scenarios leading to productivity and/or quality improvements

Model Transformations in Test Plan Automation

Unified Modeling Language

- UML is a popular modeling language widely used by software engineering professionals, especially suited to engineer MIS applications
- A semi-formal language
 - Structured but NOT a language to unambiguously specify the expected behavior of a software application
 - Offers methodologies and models to perform analysis and design of software applications

Model Based Approach to Test Plan Automation

- Generation of
 - system test , regression test and integration test
 - scenarios, cases, data and scripts

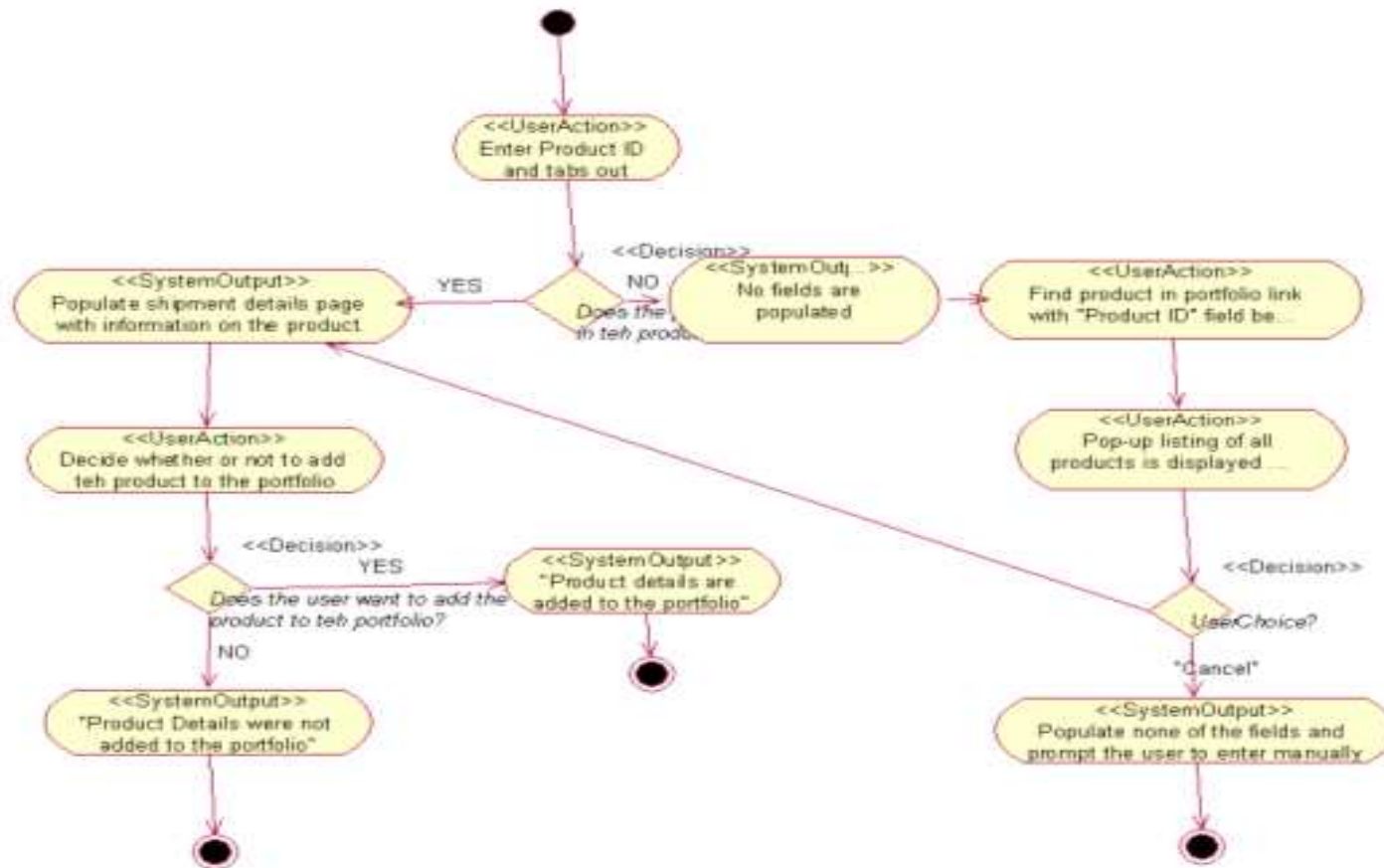
from

- Analysis models
 - UML Use-Case Activity Diagrams, State Diagrams, Communication, Collaboration Diagrams
- Design Models
 - Class Diagrams, Sequence Diagrams

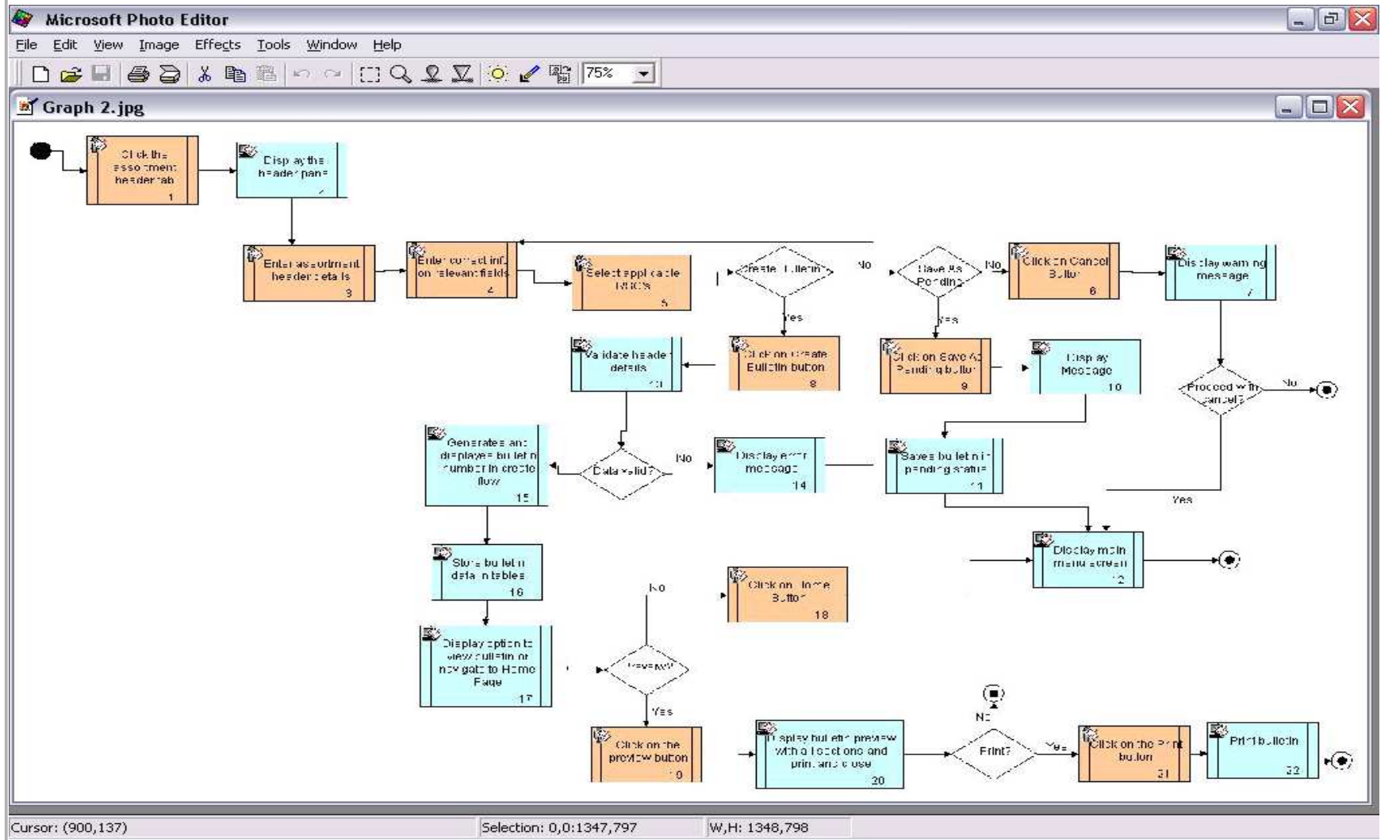
Test Case Generation

- From UML Use-Case Activity Diagrams (UCAD)
 - UCAD: Directed Cyclic Graph
 - Automatic Generation of Test Scenarios
 - One can automatically generate test scenarios through a DFS with restrictions on traversal of cyclic paths
 - Automatic Generation of Test Cases
 - One can automatically generate test cases by slicing each scenario into tuples of <Inputs, Processing, Conditions, Expected Output>

Sample UML Use Case Activity Diagram (UCAD)



Sample UML Use Case Activity Diagram (UCAD)



Test Case Generation from UCADs: Methodology

Develop one or more use-case Diagrams from requirements



Develop one or more use-case activity diagrams based on a predefined set of rules



Automatically check consistency of the use-case activity diagrams



Validate the consistent use-case activity diagrams with one or more users



Generate one or more functional test cases from the validated use-case activity diagrams

Test Case Generation from UCADs: Case Study

- Consider the following use case from ‘Automatic Teller Machine (ATM) System’: With-draw Cash using a debit-card from ATM
 - System displays the msg: “Enter the debit card number (Swipe the card)”
 - User inputs the debit-card number (swipes the card)
 - System validates the card (assume the card number is valid)
 - System displays the msg: “Enter the PIN”
 - User enters the PIN
 - System Validates the PIN (assume the PIN is valid)
 - System displays the menu of choices and the msg: “Choose from the Menu”
 - User selects the Menu option: “With-draw Cash from SB Account”
 - System displays the msg: “Enter the amount”
 - User enters the amount
 - System validates the amount (assume the amount is correct)
 - System displays the msg: “Take the amount; Thank you!”

Test Case Generation from UCADs: Case Study

- Add the following exceptions to the ATM Use Case
 - Debit-card swiped by the user is invalid
 - PIN entered by the user is invalid (the 1st time error)
 - PIN entered by the user is invalid (the 3rd consecutive time error)
 - Amount entered by the user is invalid (amount > current balance)
 - Amount entered by the user is invalid (amount > daily upper-limit)

Test Case Generation from UCADs: Case Study

Scenario #1		
Precondition(s): none		
# User Input	Conditions	Expected Output
1 Enter Product ID and tabs out	Does the product ID exist in teh product portfolio? = "NO"	1. No fields are populated
2 Find product in portfolio link with \'Product ID\' field being empty		
3 Pop-up listing of all products is displayed and select desired product and click on a link.	UserChoice? = n/a	1. Populate shipment details page with information on the product.
4 Decide whether or not to add teh product to the portfolio	Does the user want to add the product to teh portfolio? = "YES"	1. \'Product details are added to the portfolio\'

Scenario #2

Precondition(s):
none

# User Input	Conditions	Expected Output
1 Enter Product ID and tabs out	Does the product ID exist in teh product portfolio? = "NO"	1. No fields are populated
2 Find product in portfolio link with		

Results of TCG in a few Real-world Projects

	Healthcare- Claims	Internet Banking	Retail – Ecommerce
PRODUCTIVITY FIGURES FROM FIELD TRIALS			
No of test cases generated	1082	1772	23366
No of task flows involved	32	48	586
Effort estimated for manual procedure (Person Days)	23	38	465.9
Effort spent using our tool (Person Days)	13.5	17.6	132
% Effort Saving	41%	54%	72%

Test Case Generation

- From UML Use-Case Activity Diagrams (UCAD)
 - Peter Zielczynski [8] and Jim Heumann [9] offer the basics on the analysis of use-cases to manually generate test cases; good to start with.
 - Lionel Briand and Yvan Labiche [12], Clementine et al [10], Chen et al [11], Linzhang et al [13] and Chen et al [14] offer the next level of detail on semi-auto / auto generation of test cases from use-cases
 - Ravi Gorthi et al [1] use a novel concept from Paul Gerrard [15] called 'unit of behavior' to automatically generate test scenarios and test cases from UCADs

Test Case Generation

- From System Requirements Specifications (SRS)
 - SRS is typically semi-structured
 - Structured SRS
 - First break a given SRS into a set of Use-Cases
 - Express each Use-Case as ordered sequence of tuples of <Inputs, Conditions, Expected Output>
 - One can automatically generate Use-Case Activity Diagrams (UCADs) from the Structured SRS
 - Automatic Generation of Test Scenarios
 - One can automatically generate test scenarios through a DFS with restrictions on traversal of cyclic paths
 - Automatic Generation of Test Cases
 - One can automatically generate test cases by slicing each scenario into tuples of <Inputs, Conditions, Expected Output>

Structuring SRS: Use Case Template

Name	UC-01 ...		
Description	The use case has ...		
Pre-condition	...		
Interaction Steps	Input	1	The user ...
	Process (optional)	2	The system ...
	Decision (optional)	3	If (Condition is True) Goto 4 Else Goto 5
	Output	4	The system Displays ...
	Decision (optional)	5	If (...) Goto ... Else Stop

Post-Condition	...		
Non-Functional Requirements(Optional)	<Keywords such as frequency, performance, priority, fault tolerance>: Requirement Description		

ATM Example : Structuring SRS

The screenshot displays the Infosys TA Tool interface with a structured SRS for an ATM withdrawal use case. The tool window is titled "Infosys TA Tool" and contains two main text areas: a left pane for use case details and a right pane for the structured SRS.

Left Pane (Use Case Details):

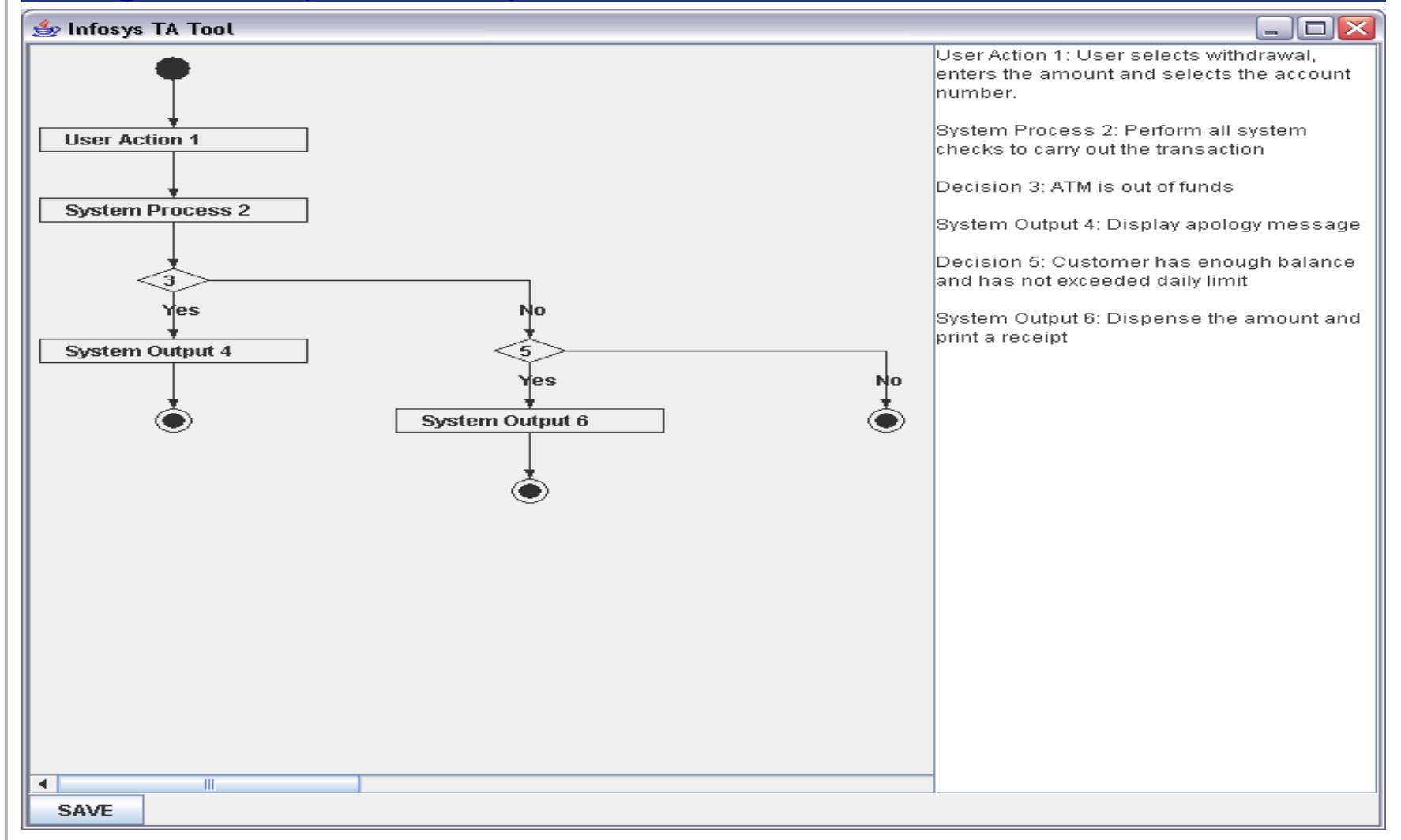
- System Name:
- User:
- Usecase Name:
- Description:
- Requirement:
- Input id="1": User selects withdrawal, enters the amount and selects
- Process id="2": Perform all system checks to carry out the transaction
- Decision id="3":
- If: ATM is out of funds
- Goto: 4
- Else:
- Goto: 5
- Decision id="5":
- If: Customer has enough balance and has not exceeded daily limit
- Goto: 6
- Else:
- Goto: stop
- Output id="6": Dispense the amount and print a receipt
- Goto: stop
- Output id="4": Display apology message
- Stop

Right Pane (Structured SRS):

- Dependency: include validatePIN use case.
- Precondition: ATM is idle, displaying a Welcome message.
- Description:
- 1. Include ValidatePIN use case.
- 2. Customer selects Withdrawal, enters the amount, and selects the accountNumber.
- 3. System checks whether customer has enough funds in the account and whether the daily limit will not be exceeded.
- 4. If all checks are successful, system authorises dispensing of cash.
- 5. System dispenses the cash amount.
- 6. System prints a receipt showing transaction number, transaction type, amount withdrawn, and account balance.
- 7. System ejects card.
- 4
- 8. System displays Welcome message.
- Alternatives:
- If the system determines that the account number is invalid, it displays an error message and ejects the card.
- If the system determines that there are insufficient funds in the customer's account, it di

At the bottom left of the tool window, there are two buttons: "New Usecase" and "SAVE".

Automatic Generation of Use Case Activity Diagrams (UCADs) from Structured SRS



Test Case Generation from Structured SRS

Scenario #1		
Precondition(s): none		
# User Input	Conditions	Expected Output
1 Enter Product ID and tabs out	Does the product ID exist in teh product portfolio? = "NO"	1. No fields are populated
2 Find product in portfolio link with \'Product ID\' field being empty		
3 Pop-up listing of all products is displayed and select desired product and click on a link.	UserChoice? = n/a	1. Populate shipment details page with information on the product.
4 Decide whether or not to add teh product to the portfolio	Does the user want to add the product to teh portfolio? = "YES"	1. \'Product details are added to the portfolio\'

Scenario #2

Precondition(s):
none

# User Input	Conditions	Expected Output
1 Enter Product ID and tabs out	Does the product ID exist in teh product portfolio? = "NO"	1. No fields are populated
2 Find product in portfolio link with		

Test Case Generation

- From UML State Diagrams / Charts
 - Philippe and Pascale [20], Supaporn and Wanchai [21] offer the basics on automatically generating of test cases from UML state diagrams
 - Stefania Gnesi et al [22] and Valdivino Santiago et all [23] offer the next level of details on the automated generation of test cases from UML state diagrams
 - Jeff Offutt et al [24] and Stefan Hildenbrand [25] offer great details on the analysis of state-based specifications and finite state automata to generate test cases.

Model Based Regression Test Case Selection

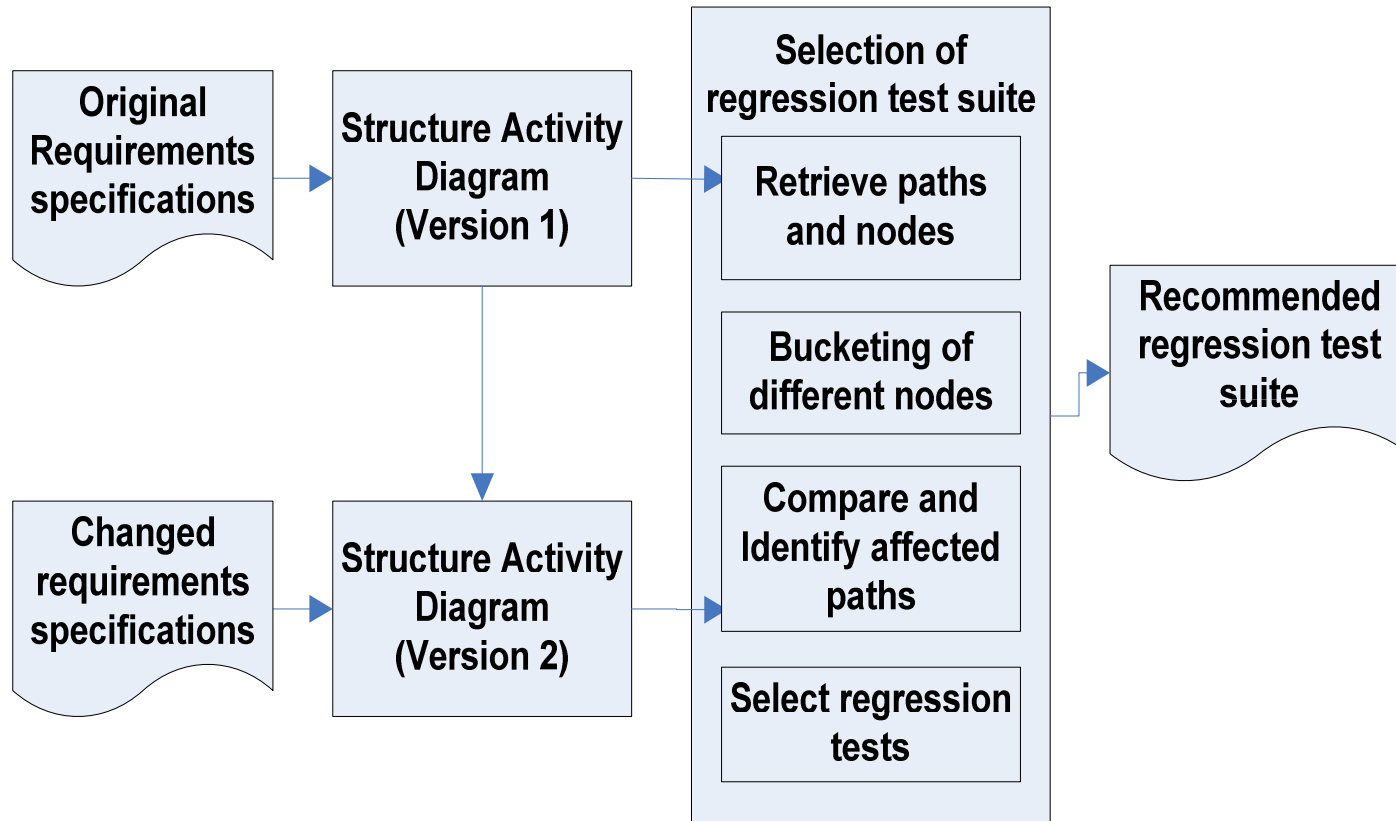
Some important issues related to regression testing:

- When a software application is modified due to ‘change requests’ received from business users, the application need to be tested to ensure that the modified system meets all the functional (and non-functional) requirements
- Typically, medium to large software applications are observed to involve,
 - a few tens of to a few hundreds of ‘o-o classes’
 - A few thousands of to a few tens of thousands of ‘methods’
- In general, each cycle of software maintenance involves changes to around 15 to 30% of methods
- There will be many classes and methods which remain unchanged across maintenance cycles
- Do we need to test a modified application, **completely**? Answer is **NO!**

Model Based Regression Test Case Selection

- Input
 - Two *consecutive versions* of UML Use-Case Activity Diagrams (UCADs)
 - **UCAD-v1 and UCAD-v2**
- Processing
 - *Identify* changed nodes in UCAD-v2 with respect to UCAD-v1
 - **Modified, newly added, deleted or shifted nodes**
 - *Select* regression test suite
 - **Paths in the UCAD-v2 that are affected due to the above types of node changes**
 - *Paths* in UCAD-v2 that are the same as those in UCAD-v1 need *NOT be tested*
- Output
 - *Selected regression test suite* to test UCAD-v2

Model Based Regression Test Case Selection

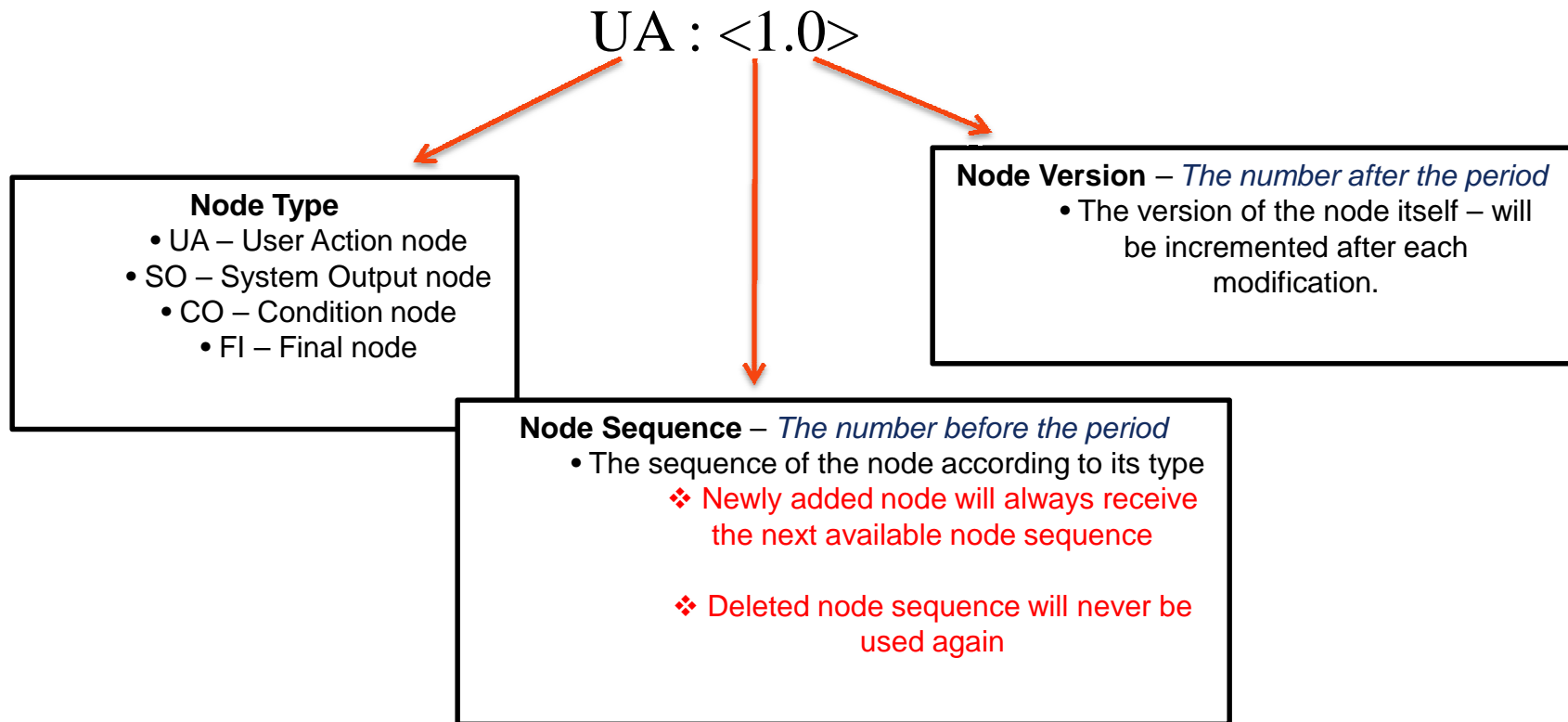


Model Based Regression Test Case Selection

- **Identify** changes to the nodes of UCAD-v2 with respect to UCAD-v1
 - Nodes in a given UCAD are structured as
 - **Node type <Sequence-number.Version-number>**
 - **E.g UA <1.0>; SP <2.3>; CO <3.1>; SO <1.4>**
 - All changes to the nodes of UCAD-v2 can be categorized as
 - **Modifications to an existing node**
 - **Deletion of an existing node**
 - **Shifting of an existing node**
 - **Addition of a new node**
- Generate **Selected** Regression Test Suite for UCAD-v2
 - Paths that contain one or more **changed** nodes

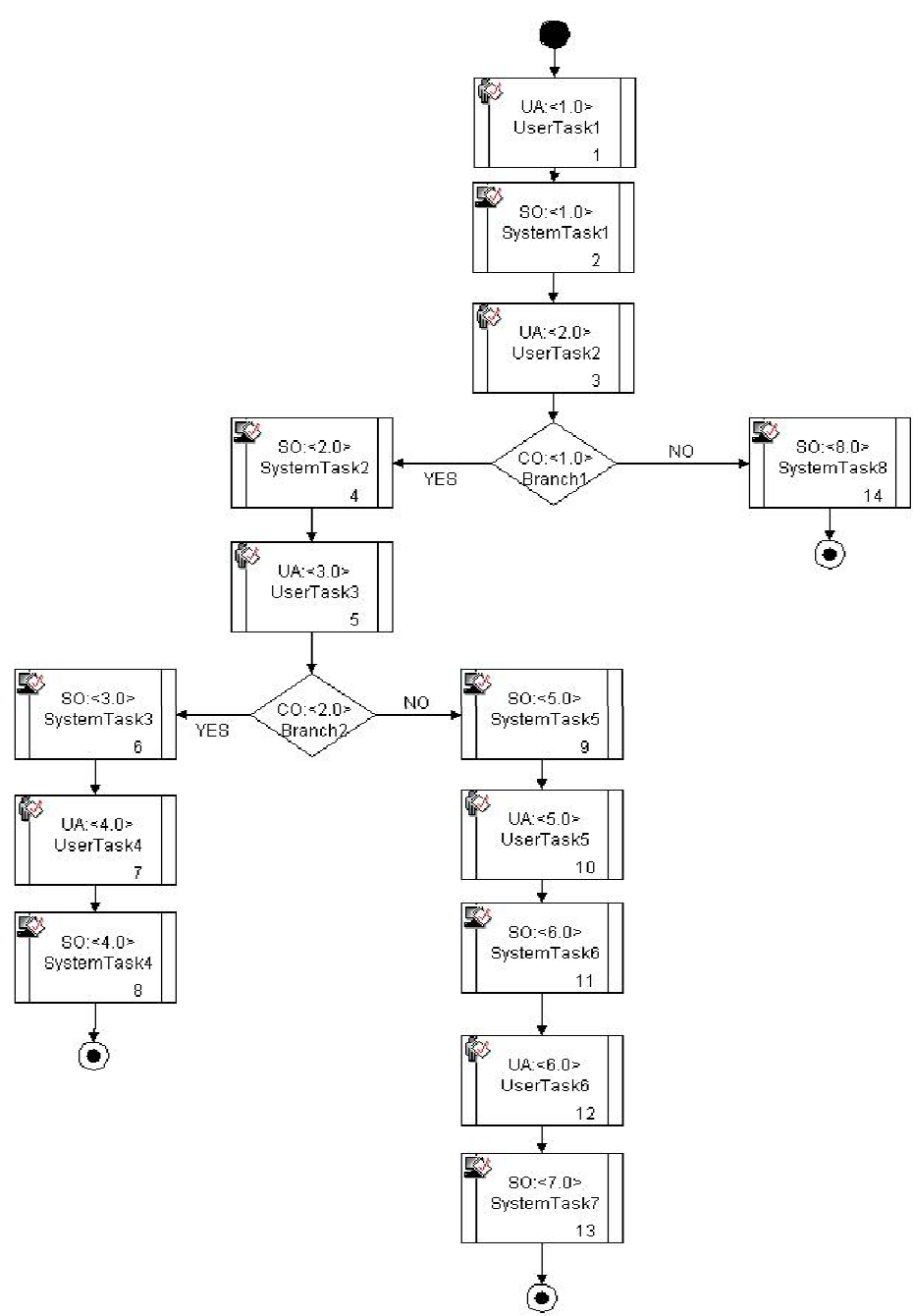
Model Based Regression Test Case Selection

- Examples : UA: <1.0> , SO: <2.2>, CO: <3.0>, FI: <1.0>
- Structure of Node Version Number



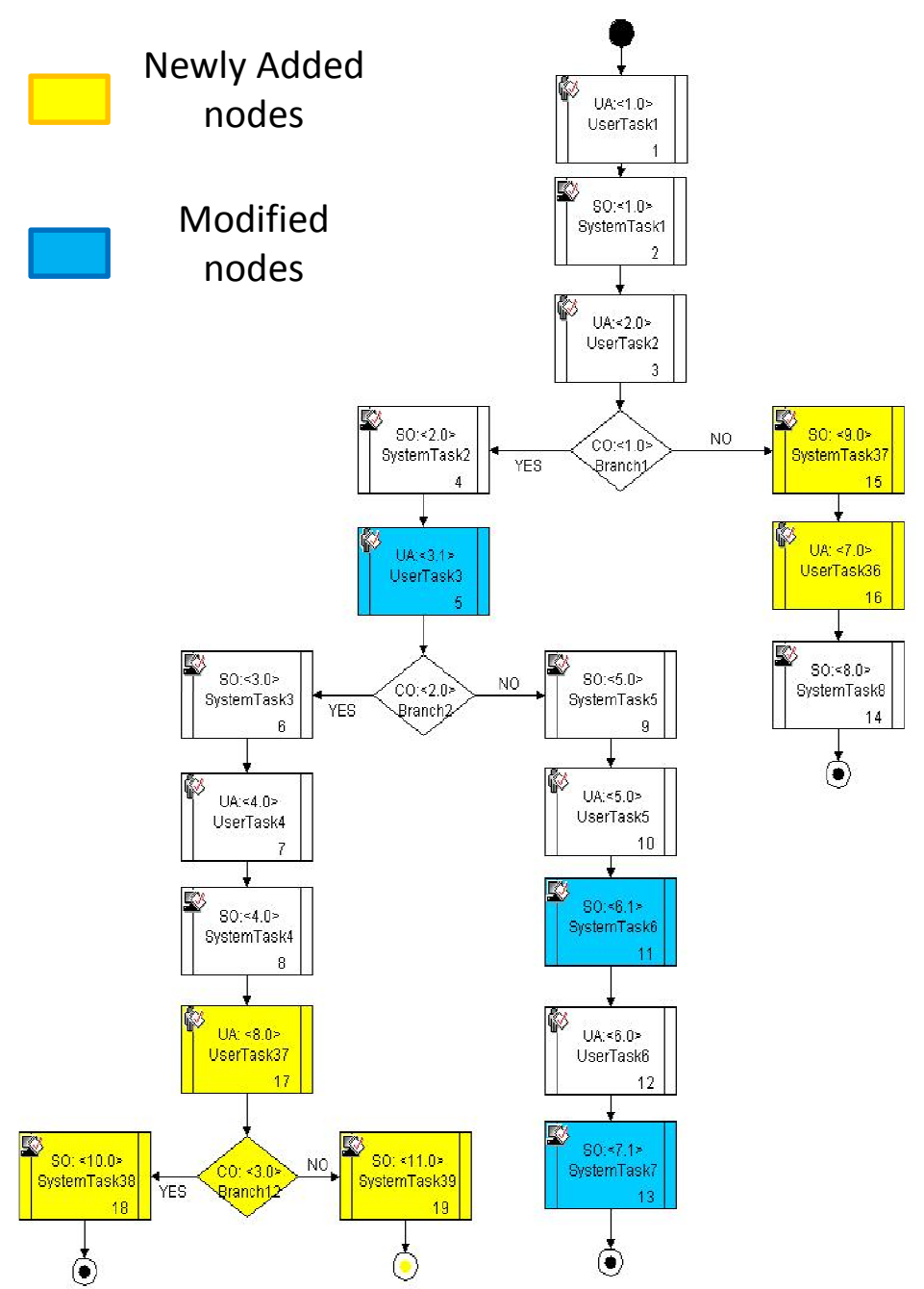
Model Based Regression Test Case Selection

- Some nodes are more critical than the others.
- Example: Verification of PIN number and account balance VS. Verification of correct display of welcome message
- Assign criticality to nodes
 - [H] – High criticality
 - [M] – Medium criticality
 - [L] – Low criticality (Default setting)



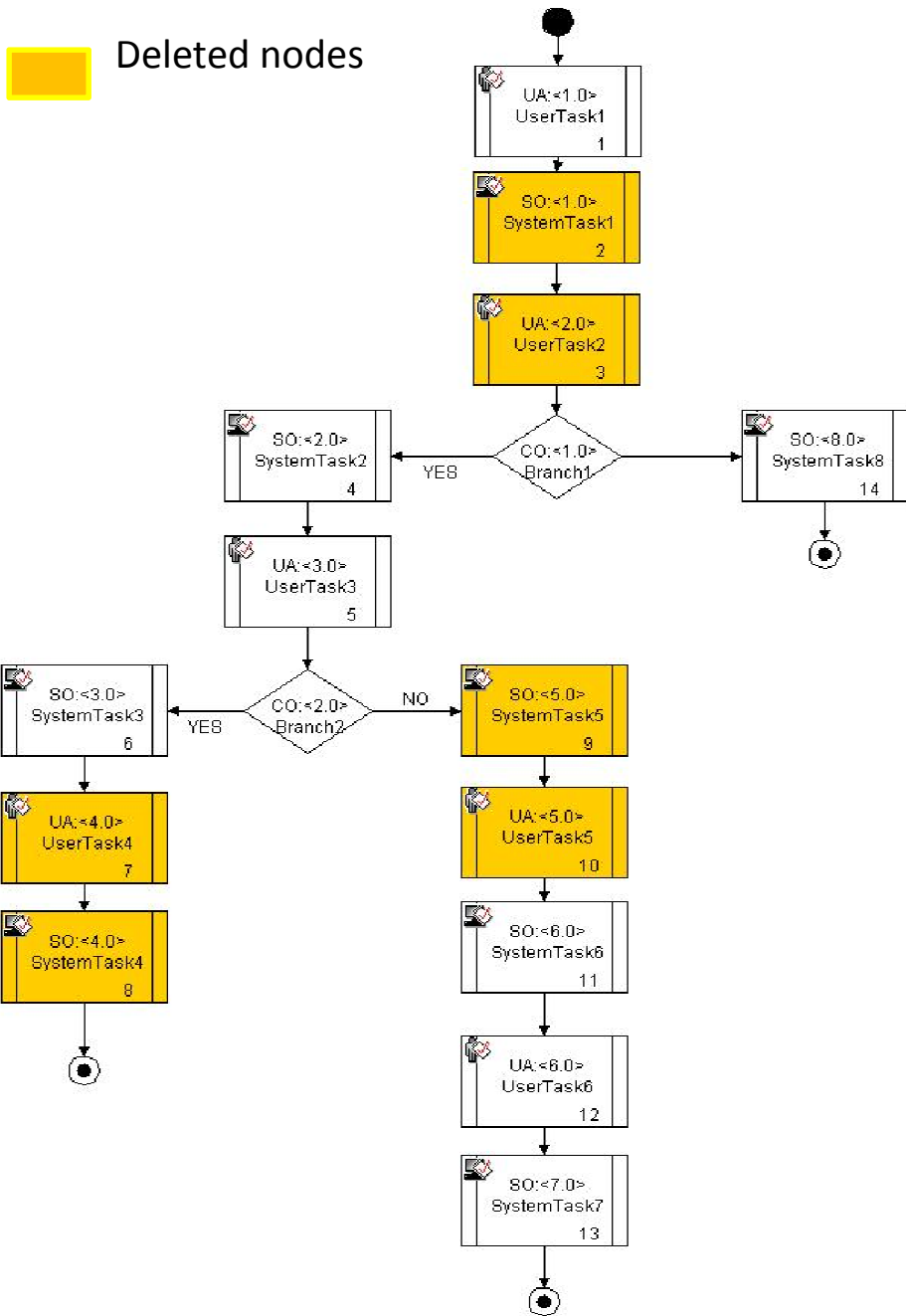
Original Influx Taskflow Diagram

Newly Added nodes
Modified nodes

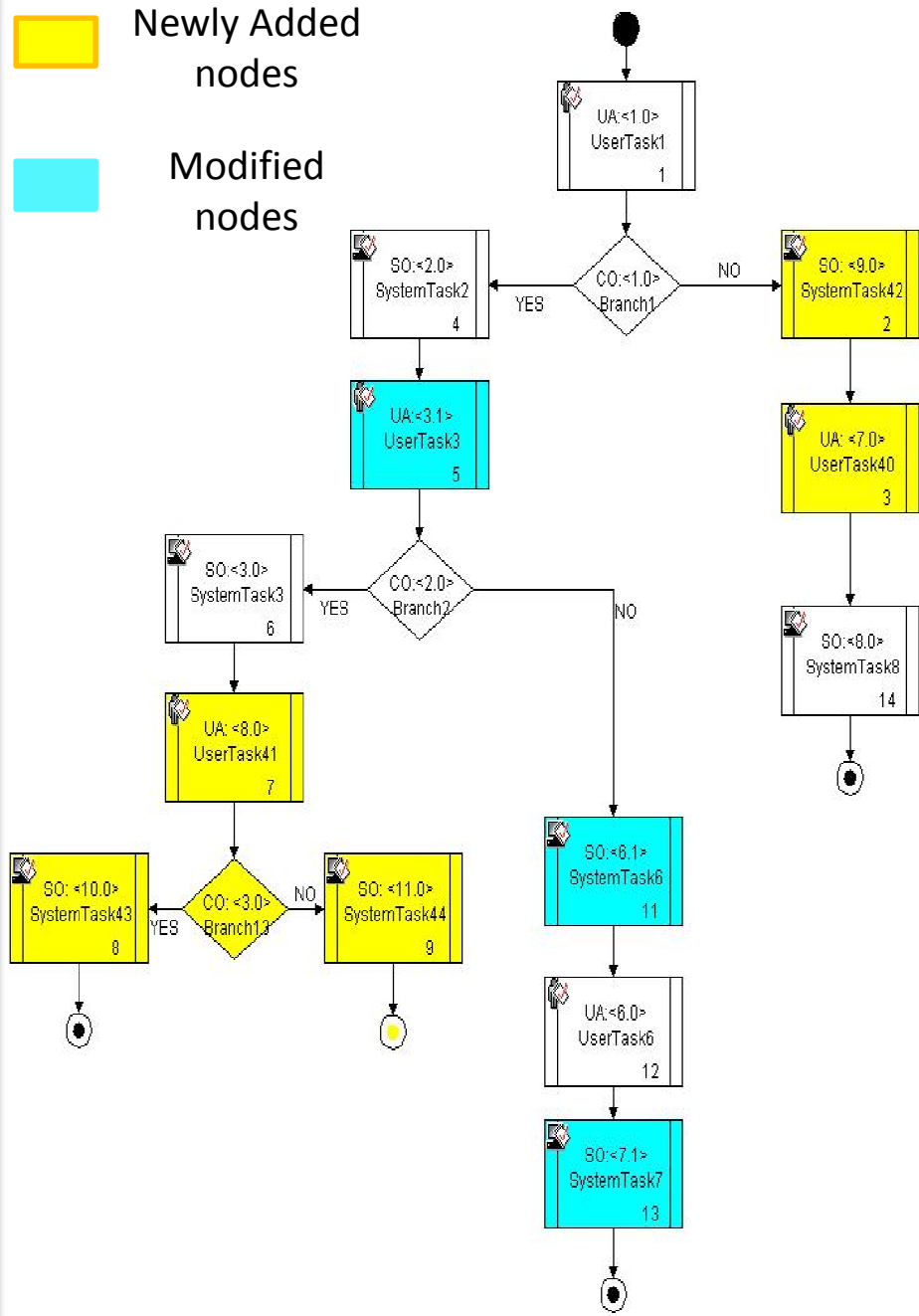


New Influx Taskflow Diagram

Deleted nodes



Newly Added nodes
Modified nodes



Original Influx Taskflow Diagram

New Influx Taskflow Diagram

Model Based Regression Test Case Selection

Description	New ID	Description	Status
Enters the operation to be performed	N/A	N/A	Deleted
Presses button to Withdraw cash	N/A	N/A	Deleted
Test Steps	Input	Conditions	Expected Output
	1 Enter the card	Card inserted properly? = Yes	1. Display a message to enter the PIN
	2 Enter the PIN	Cash withdrawal? = yes	1. Display a message to continue with the trans:
	3 Enter the amount	Cash available? = Yes	1. Display a message to collect the cash
Description	New ID	Description	Status
Enters the operation to be performed	N/A	N/A	Deleted
Presses button to Withdraw cash	N/A	N/A	Deleted

Experimental Results from a few Real-World Projects

	Healthcare-Claims	Internet Banking	Retail Order Management
No of UCAD paths involved	320	730	4300
No of regression test cases selected	1610	4780	20410
Effort estimated for manual procedure (Person Days)	23	78	466
Effort spent using our tool (Person Days)	13.5	37.6	186
<i>Productivity gain</i>	<i>41%</i>	<i>52%</i>	<i>60%</i>

Selective References

1. Ravi Gorthi et al., “Model-Based Automated Test Case Generation”, SETLabs Briefings, Vol 6, No 1, 2008, pp 39 – 46. <http://www.infosys.com/research/publications/toc-software-validation.asp>
2. Ravi Gorthi, Anjaneyulu Pasala, et al., Specification-based Approach to Select Regression Test Suite to Validate Changed Software, accepted for publication at the 15th Asia-Pacific Software Engineering Conference (APSEC - 2008), Dec. 3-5, 2008.
3. Anjaneyulu Pasala, Ravi Gorthi, et al., “Selection of regression test suite to validate software applications upon deployment of upgrades”, 19th Australian Software Engineering conference (ASWEC – 2008), 25-28 March 2008, pp 130-138.
4. Anjaneyulu P, et al. “An Approach for Test Suite Selection to Validate Applications on Deployment of COTS Upgrades”, *12th IEEE Asia Pacific Software Engineering Conference*, Dec. 2005, pp 401–407.
5. Anjaneyulu P, et al “An Approach Based on Modeling Dynamic Behavior of the System to Assess the Impact of COTS Upgrades”, 13th IEEE Asia-Pacific Software Engineering Conference (APSEC), Dec. 2006, pp 19-26.
6. Anjaneyulu Pasala, Ravi Gorthi et al., “How to Select Regression Tests to Validate Applications upon Deployment of Upgrades”, SETLabs Briefings, Vol 6, No1, 2008, pages 55 – 63.
7. Anjaneyulu Pasala et al., “On the validation of API execution-sequence to assess the correctness of application upon COTS upgrades deployment”, 6th IEEE International conference on COTS based software systems, (ICCBSS - 2007), February-March 2007, at Banff, Canada.
8. Peter Zielczynski, “Traceability from Use Cases to Test Cases”, <http://www.ibm.com/developerworks/rational/library/04/r-3217/>

Selective References

9. Jim Heumann, “Generating Test Cases from Use Cases”, the Rational Edge, June 2001
10. Clementine Nebut, et al, “Automatic Test Generation: A Use Case Driven Approach”, IEEE Transactions on Software Engineering, vol 32, n 3, March 2006
11. Chen T Y, et al, “A Choice Relation Framework for Supporting Category-Partition Test Case Generation”, IEEE Transactions on Software Engineering, vol 29, no 7, July 2003
12. Lionel Briand and Yvan Labiche, “A UML-Based Approach to System Testing”, Carleton University TR SCE-01-01-Version 4, June 2002
13. Wang Linzhang, et al, “Generating Test Cases from UML Activity Diagram based on Gray-Box Method”, Proceedings of APSEC 2004
14. Chen Mingsong, et al, “Automatic Test Case Generation for UML Activity Diagrams”, AST 06, Shanghai, China, May 2006
15. Paul Gerrard, “Testing Requirements”, Systeme Evolutif, <http://www.evolutif.co.uk/testReqs/TESTREQS.html>
16. Thomas Ostrand and Marc Balcer, “The Category Partition Method for Specifying and Generating Functional Tests”, Communications of the ACM, vol 31, no 6, June 1988
17. Aynur Abdurazik and Jeff Offutt, “Generating Test Cases from UML Specifications”, ISE-TR-99-09, Information and Software Engineering, George Mason University, May, 1999
18. James Clarke, “Automated Test Generation from a Behavioral Model”, Software Quality Week Conference, May, 1998

Selective References

19. James Clarke, “Automated Test Generation from a Behavioral Model”, Software Quality Week Conference, May, 1998
20. Philippe Chevalley and Pascale Thevenod-Fosse, “Automated Generation of Statistical Test Cases from UML State Diagrams”, Proceedings of COMPSAC 2001
21. Supaporn Kansomkeat and Wanchai Rivepiboon, “Automated-Generating Test Case Using UML Statechart Diagrams”, Proceedings of SAICSIT, 2003
22. Stefania Gnesi et al, “Formal Test-Case Generation for UML Statecharts”, Proceedings of 9th IEEE Int Conf on Engineering Complex Computer Systems, 2004
23. A-Valdivino Santiago et al, “A Practical Approach for Automated Test Case Generation using Statecharts”, Proceedings of COMPSAC 2006
24. Jeff Offutt et al, “Generating test data from state-based specifications”, in Software Testing, Verification and Reliability, John Wiley & Sons Ltd, 2003
25. Stefan Hildenbrand, “Generation of Test Cases”, Semester Thesis, Summer of 2005, ETH Zurich
26. Basanieri F and Bertolino A, “A Practical Approach to UML-based derivation of integration tests”, <http://www1.isti.cnr.it/~antonia/publications2002.html>
27. Richard DeMillo and Jefferson Offutt, “Constraint-Based Automatic Test Data Generation”, IEEE Transactions on Software Engineering, vol 17, no 9, Sep 1991
28. Bogdan Korel, “Automated Software Test Data Generation”, IEEE Transactions on Software Engineering, vol 16, no 8, August 1990

Selective References

29. Gregg Rothermel et al., “a safe, efficient regression test selection technique”, ACM transactions on software engineering and methodology, Vol. 6, No. 2, April 1997, pp 173 – 210.
30. Greg Rothermel, et al., “Analyzing Regression Test Selection Techniques”, IEEE Transactions on Software Engineering, Vol. 22, No. 8, August 1996, pp 529-551.
31. Zheng J, et al. “Applying Regression Test Selection for COTS-based Applications”, International Conference on Software Engineering (ICSE), May 20-28, 2006, pp 512-521.
32. Briand, L.C.; Labiche, Y.; Soccar, G., “Automating impact analysis and regression test selection based on UML designs”, International Conference on Software Maintenance, 3-6 October 2002, pp 252 – 261.