Eiropas Sociālā fonda projekts
“Datorzinātnes pielietojumi un tās saiknes ar kvantu fiziku”
Nr.2009/0216/1DP/1.1.1.2.0/09/APIA/VIAA/044

SIMPLIFIED DESIGN OF TEST CASES BASED ON MODELS

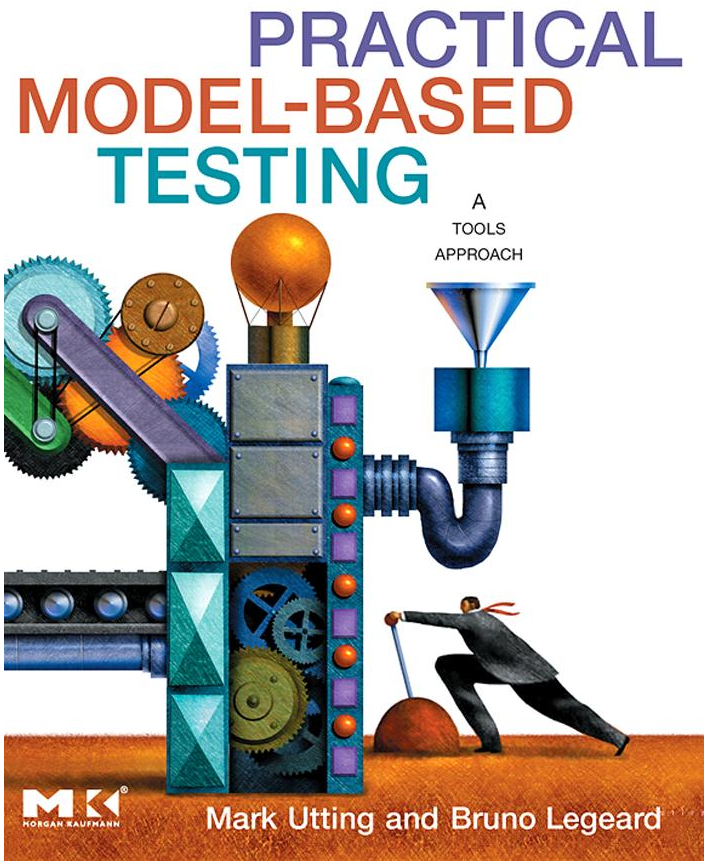
Guntis Arnicans, Vineta Arnicane

26.05.2011

12th International Conference Theory and Practice of Software Testing (TAPOST 2011)
"Forming Basis of Globally Mature Testing"

Model-based testing

1. A formal model that describes the behavior of the software
2. A test generation algorithm or criteria
3. Tools that support a test infrastructure (test cases generation, management, implementation, evaluation of results, etc.)



Problems with model-based testing

- Most of the steps have to be automated
- Model have to be described by formal notation (e.g. UML, Finite state machines)
- The need to have good knowledge of formal testing, lack of good specialists
- Difficulty to translate data generated by the model to final test cases
- Model of the whole system is welcomed
- Natural complexity (at least the same as of the source code)

Most of model-based testing approaches are not empirically verified and used in the industry

Case study from insurance

- Tariffication module for a compulsory motor third party liability insurance
- Simplified model in Excel
- Test case generation from Excel model
- Running test cases with tariffication module developed in Oracle and Excel
- If results of test case are different then manual investigation is performed
- More than 50’000 automated test cases
- Model was used and testing was automated due the importance of this module (expensive, but worth)

Design of test cases from models

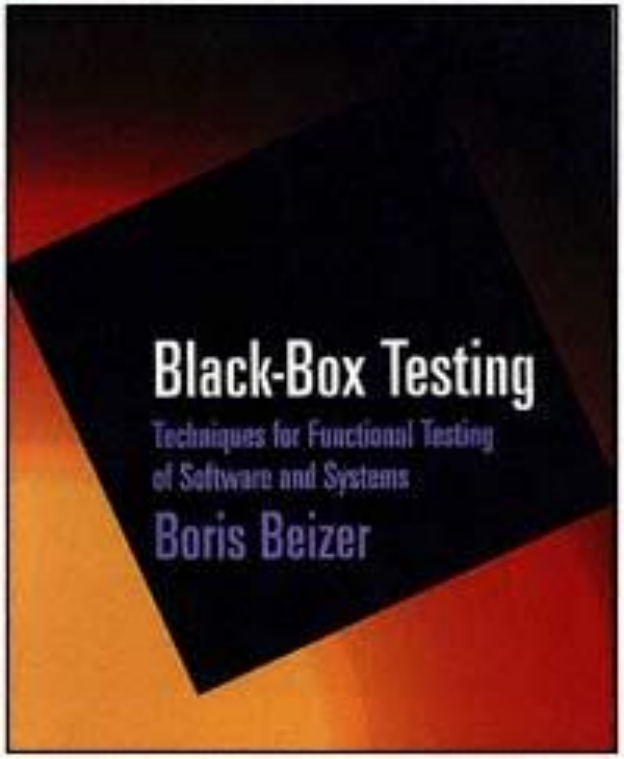
Pick up from the model-based testing approach the simplest ideas and techniques, and use them at least in the design of test cases

- Develop several small models
- Any model can be described by its own informal notation
- Recreate models and test case specifications

Use graphs with any notation

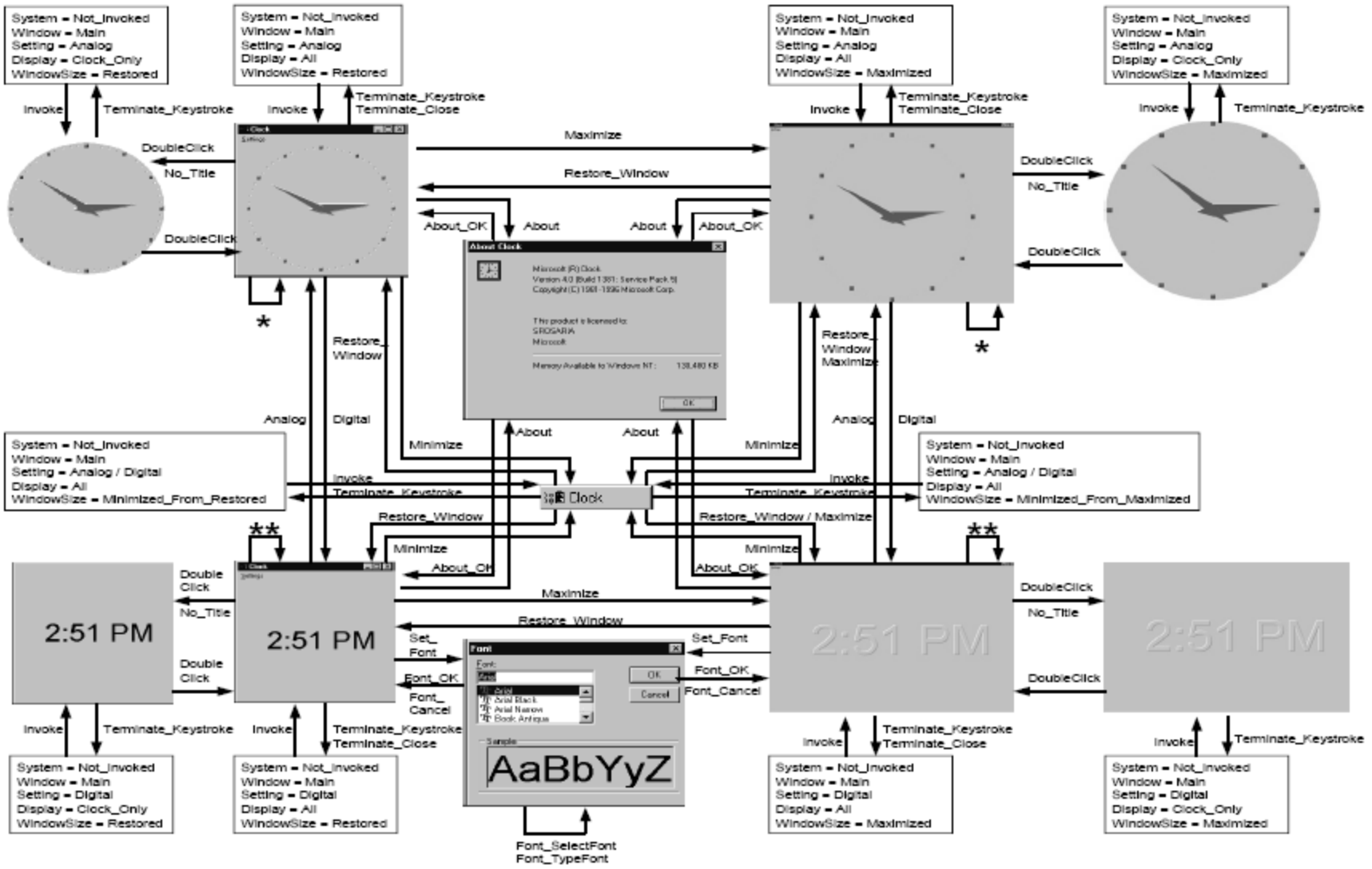
These are the steps to use a graph model to design test cases:

1. *Define the graph.*
2. *Define the relation.*
3. *Design node-cover tests (tests that confirm that the nodes are there).*
4. *Design link-cover tests (that confirm all required links and no more).*
5. *Test all weights.*
6. *Design loop tests.*



Boris Beizer, Black-box Testing : Techniques for Functional Testing of Software and Systems

Microsoft Windows® clock application



From S. Rosaria, H. Robinson, Applying Models in your Testing Process

Many models

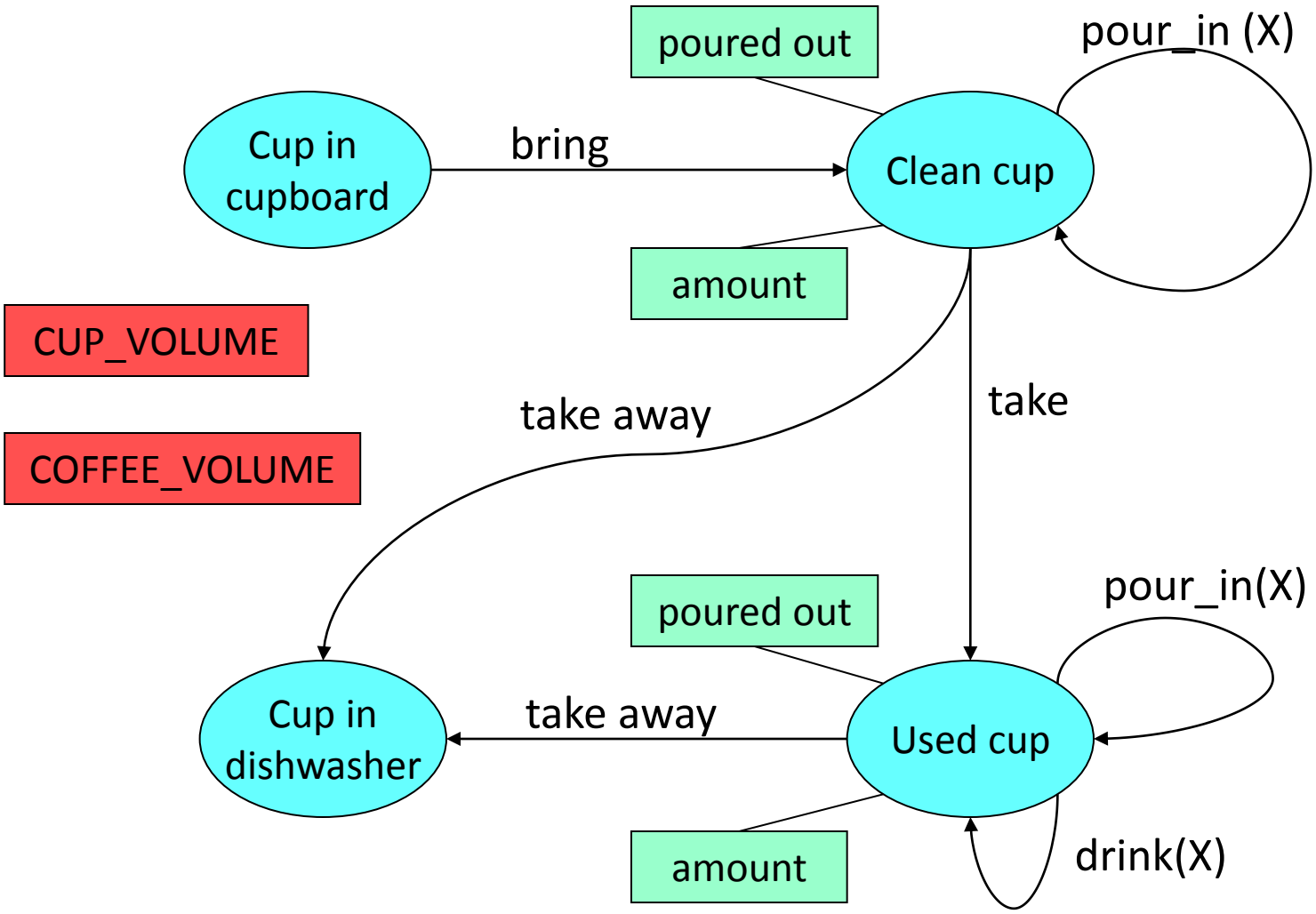
- Create many small models instead of one large
  - preferably no more than 10 main objects in a model
- Create models from the various aspects
  - different points of view
  - diverse testing can reveal different problems
- Create several models for the chosen aspect
  - each tester has own viewpoint to the task
  - various opinions what software has to do

If an inconsistency between test cases is discovered, then additional analysis and investigation have to be done

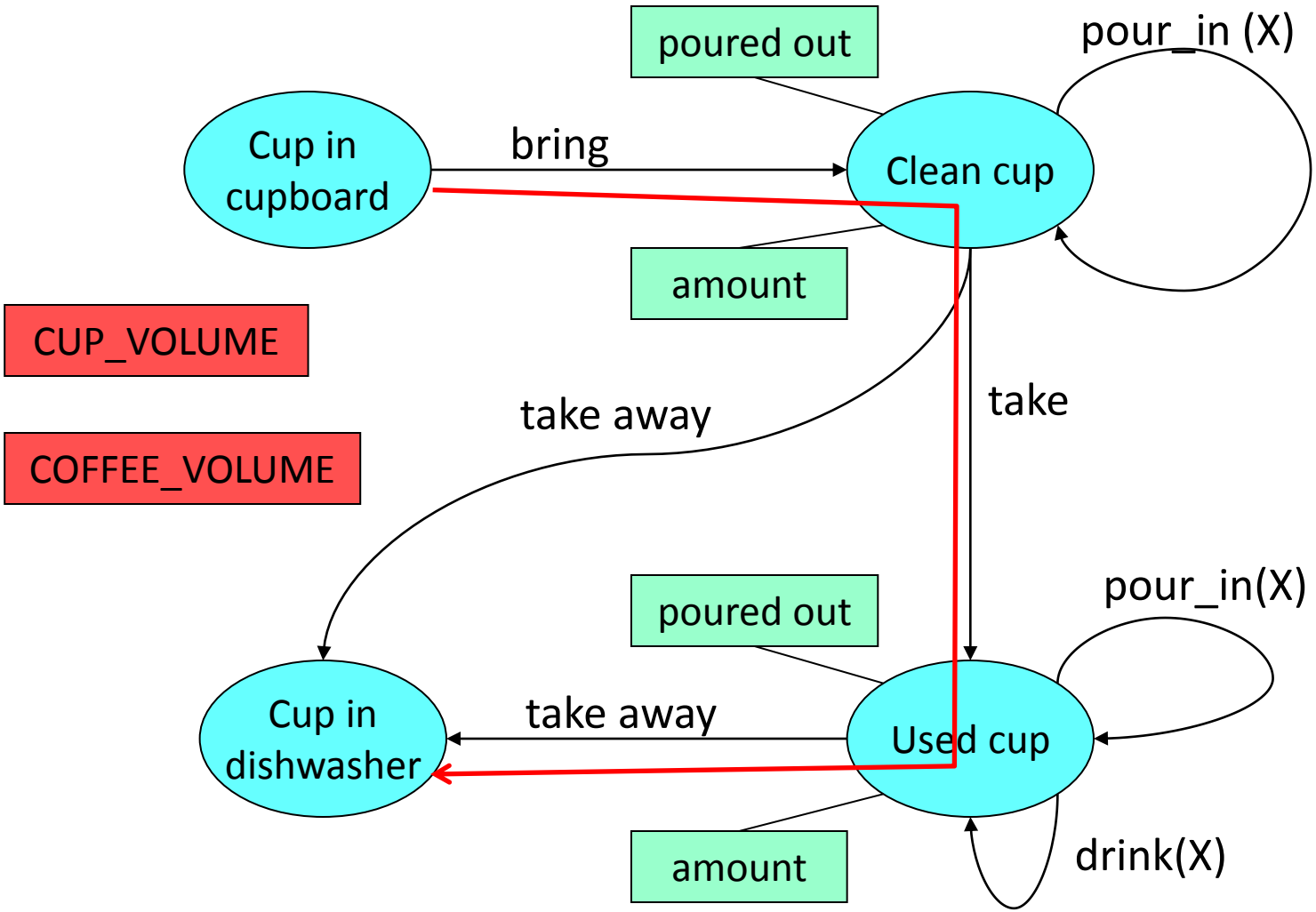
Test generation is based on well known graph coverage criteria

- Tester can adapt any coverage criteria from the structural testing
  - node coverage
  - node pair coverage
  - edge coverage
  - in-out edge pair coverage for node
  - path coverage with or without cycles
  - etc.
- If some node or edge has attributes and constraints, then principles of equivalence class testing and boundary testing methods could be applied

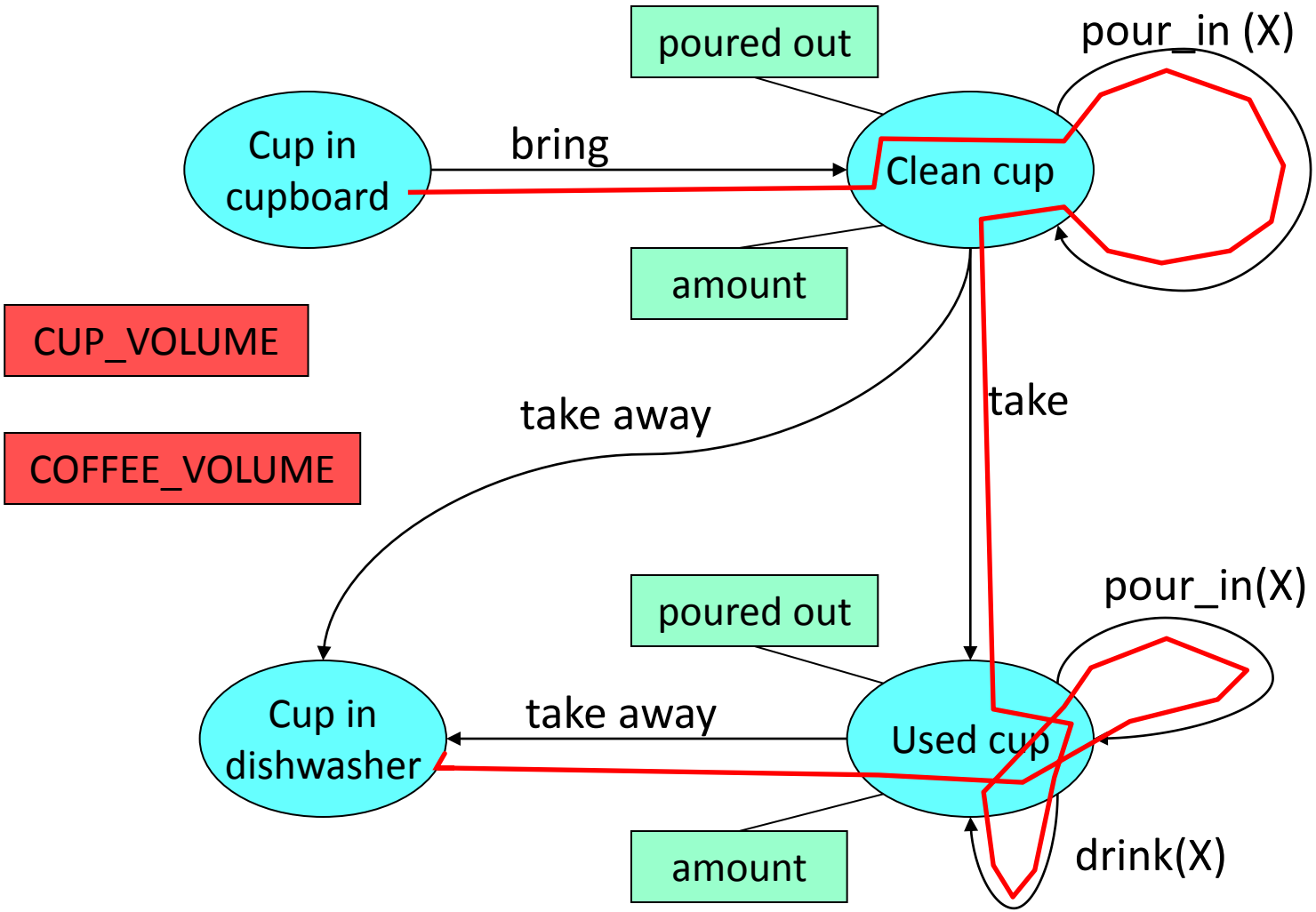
Coffee break model



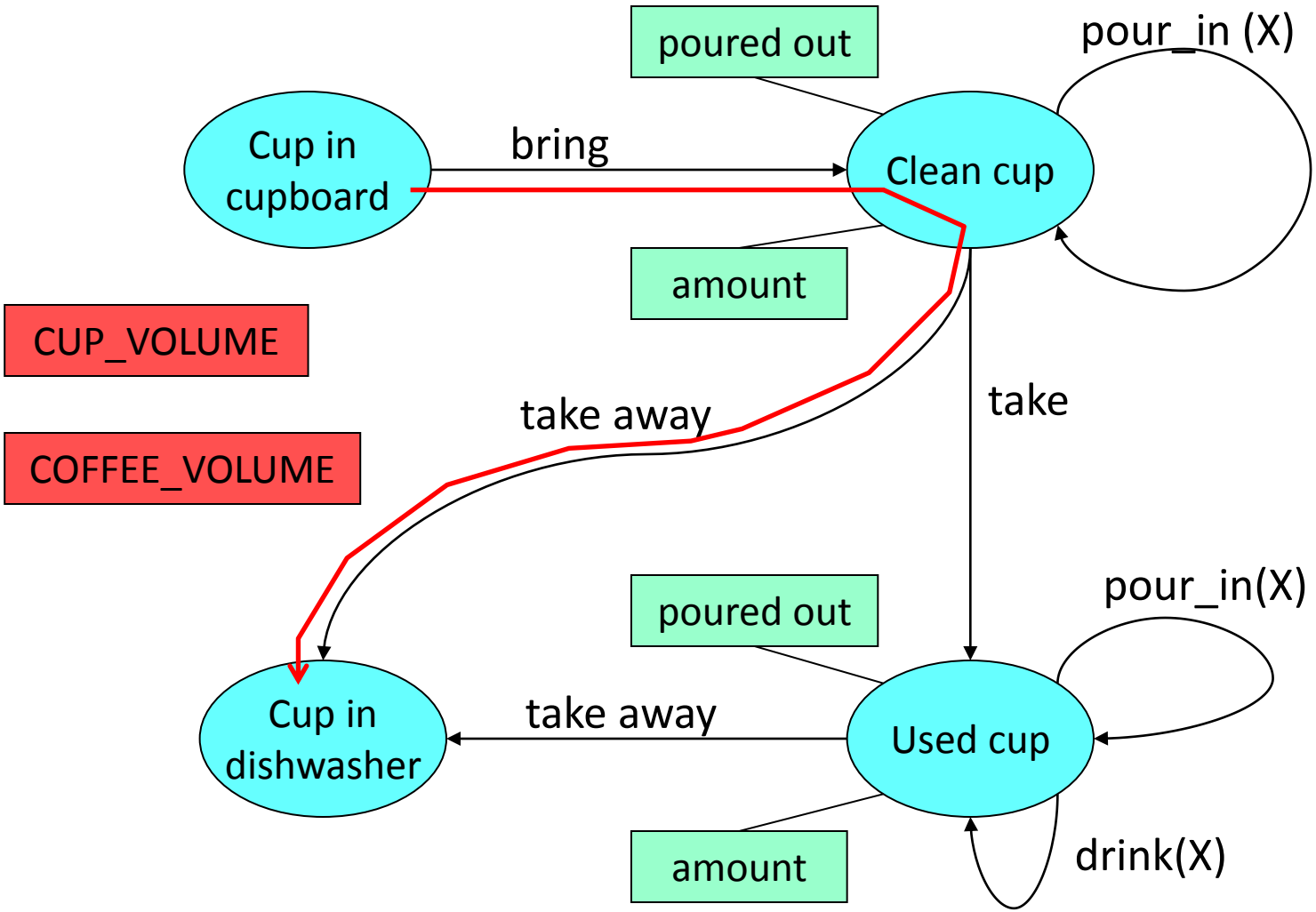
Coffee break model



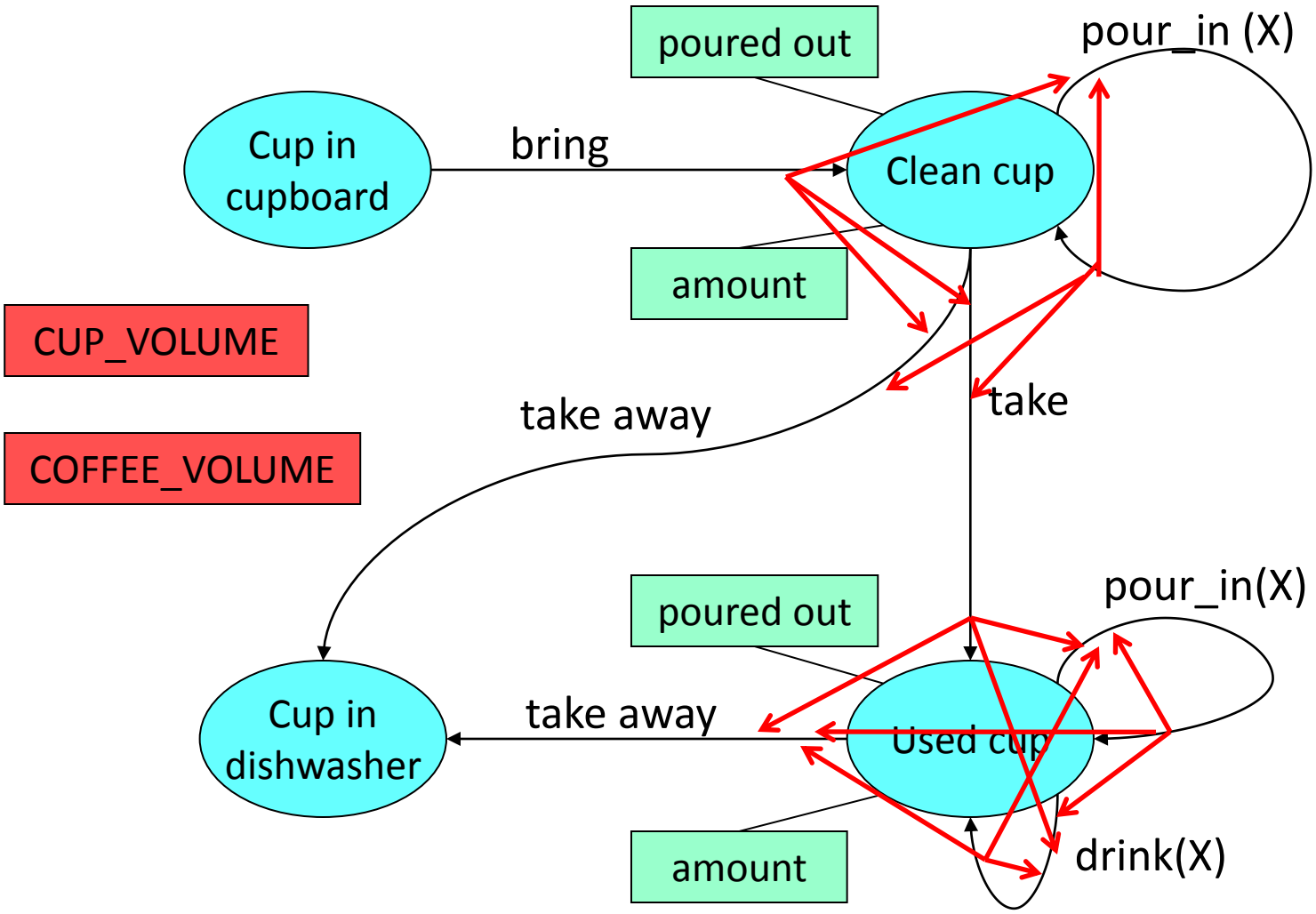
Coffee break model



Coffee break model



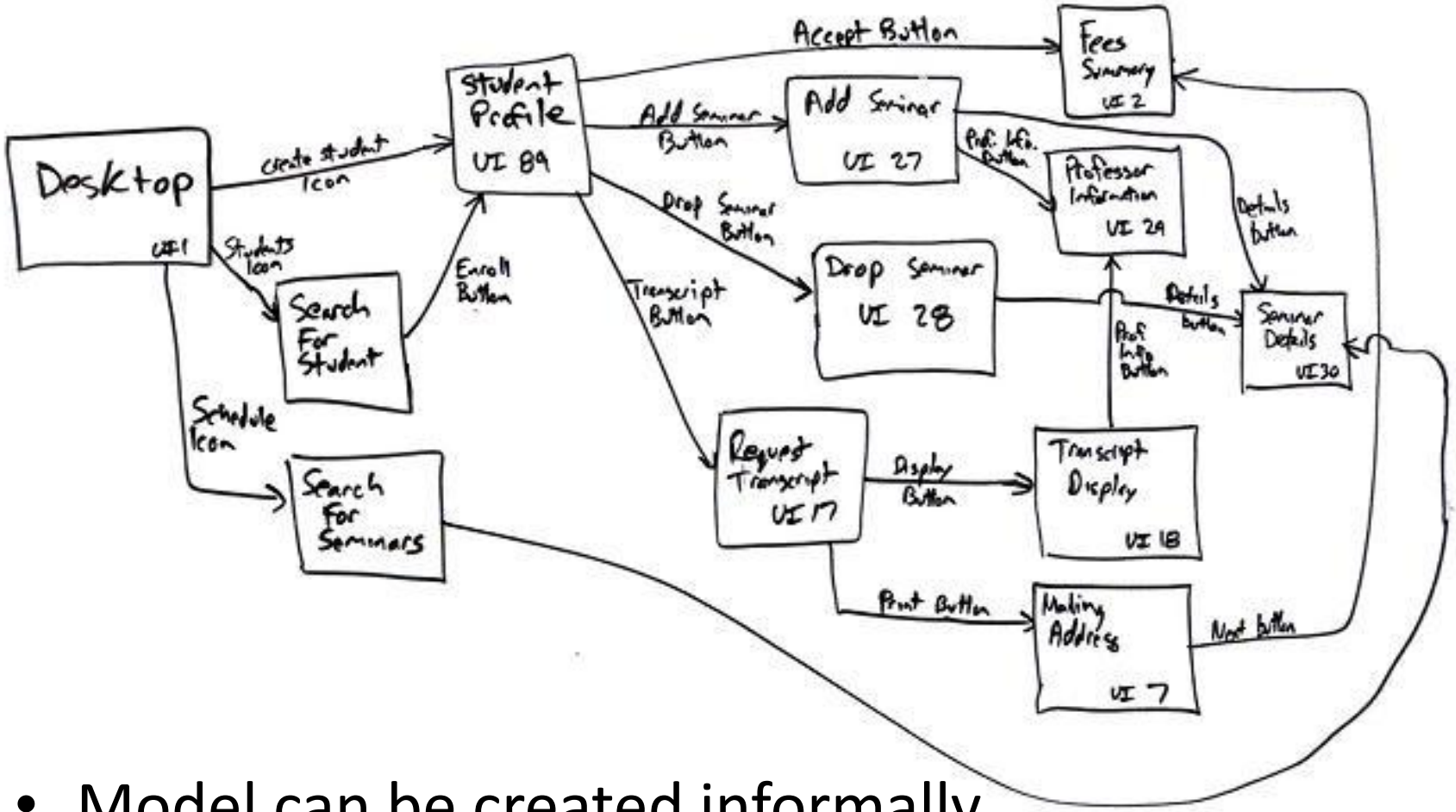
Coffee break model



Breadth testing principle to design and organize the test cases

- We have many different small models
- From each one we can design test cases
- At first we advise to execute tests that are easily obtainable and cover many aspects
- A sample:
  1) 90% node coverage in each model
  2) 90% edge coverage in each model
  3) 90% edge pair coverage for each node in each model
  4) testing of cycles
  5) testing of attributes
  6) other criteria.

Informal models



- Model can be created informally
- Hand-drawn model also is worth
- Model may be incomplete

Other principles

- Models are created for various levels of abstraction
- Test cases are designed at logical level
- Create library of exploited models and patterns
- Try and throw away

Conclusions

- The proposed method might be useful for exploratory testers
- Models help to design test cases, organize them
- Models reveal the most important issues of system from various points of view
- Use models as you can and like (without formal notations, without automation)