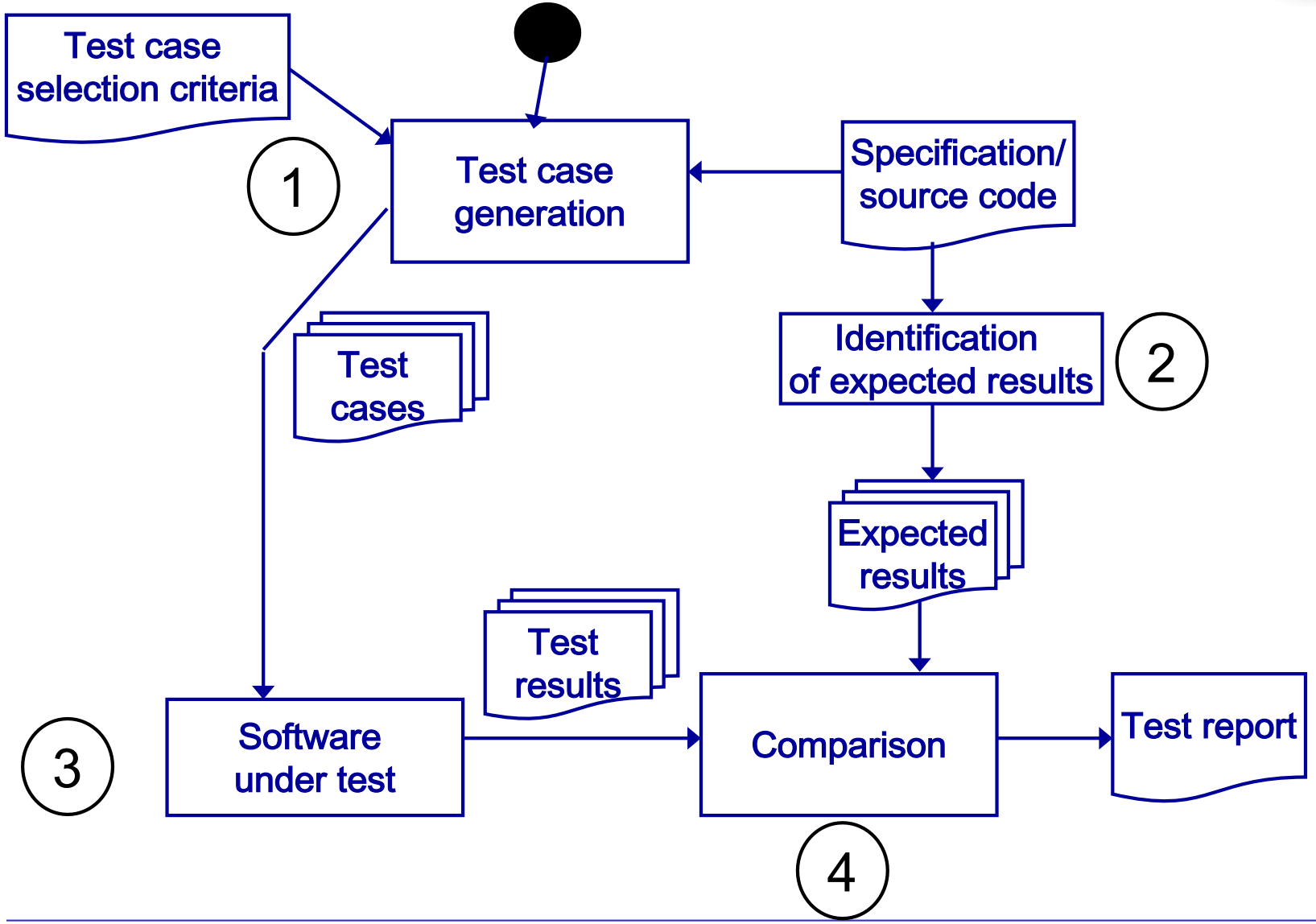
Politecnico
di Milano

Short Overview of Some Testing Techniques

The testing process

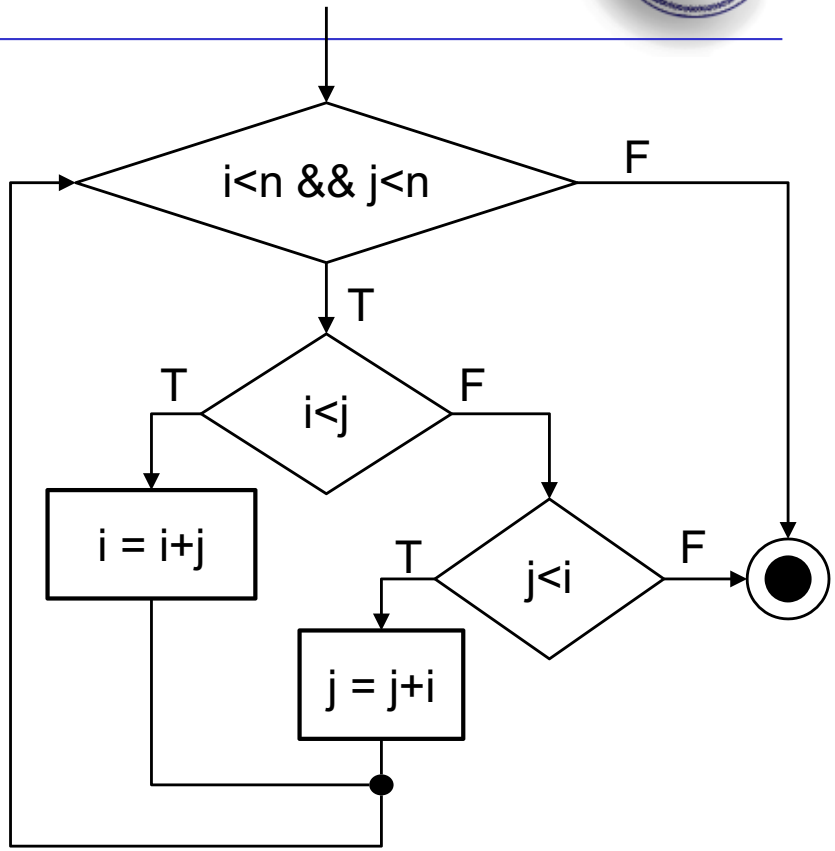


Black-box vs white-box systematic testing

- White-box testing is suitable for unit testing
  - Covering a small portion of software is possible
- Black-box testing is suitable for integration, system and acceptance testing
  - Specs usually smaller than code
  - Specs help identifying missing functionalities in the system

White-box testing: an example

```
1  //read i,j,n
2  while (i<n && j<n) {
3    if (i < j)
4      i = i + j;
5    else if (j < i)
6      j  = j + i;
7    else break;
8  }
9  //It continues...
```



(a) Define a test set that satisfies the *edge* criterion (R: ⟨3, 3, 10⟩; ⟨3, 4, 10⟩)

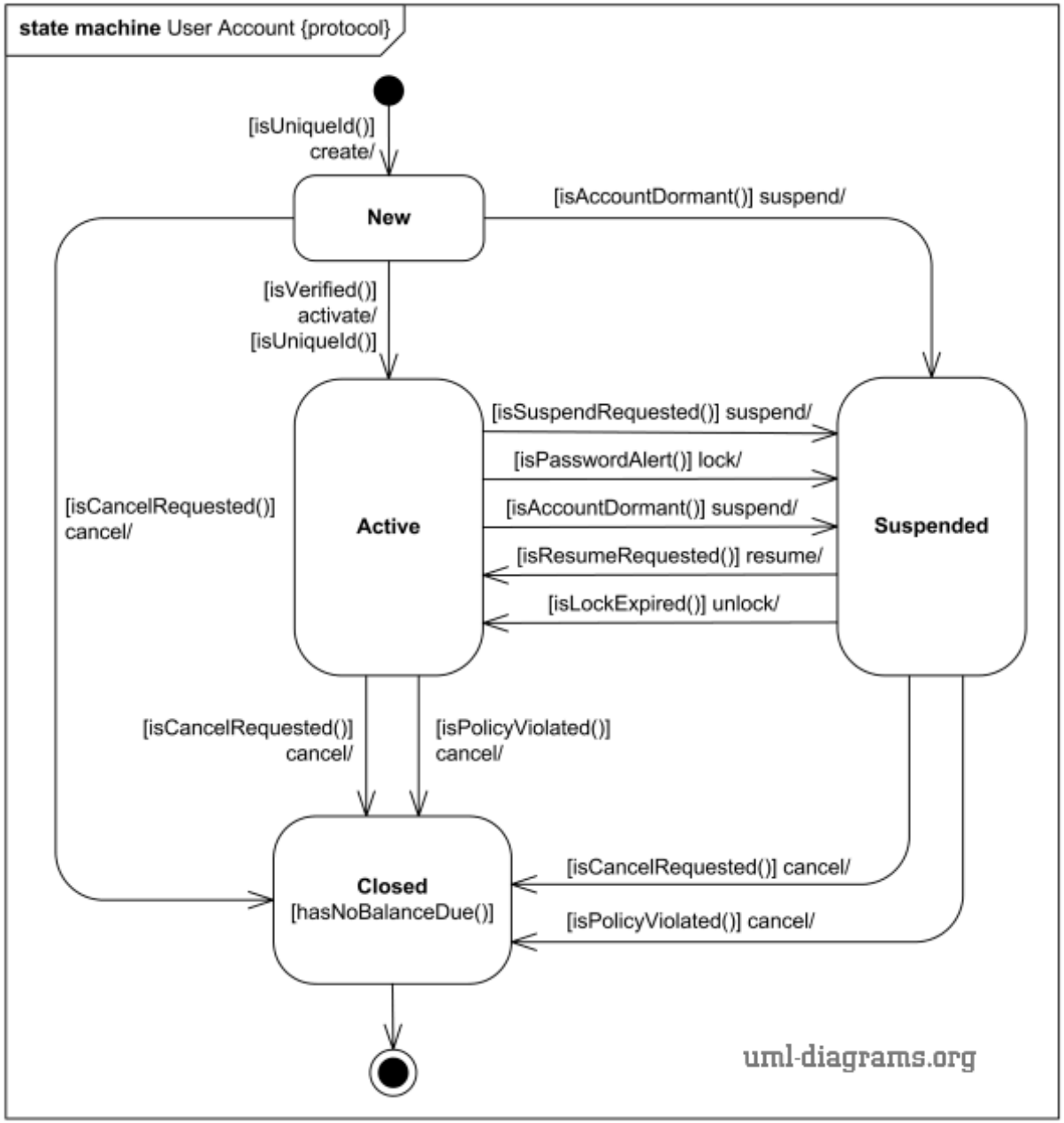
(b) Define a test set that satisfies the *condition* criterion (R: ⟨3, 3, 10⟩; ⟨3, 4, 10⟩; ⟨4, 3, 10⟩)

(c) Define a test suite that covers the following *path*: 1, 2, 3, 4, 8, 2, 3, 5, 6, 8, 2, 9 (R: ⟨3, 4, 10⟩)

An example of black-box testing: Model-Based Testing (MBT)

- We can use models to devise test cases
- The actual behavior of the software under test are checked against the behavior specified by the model

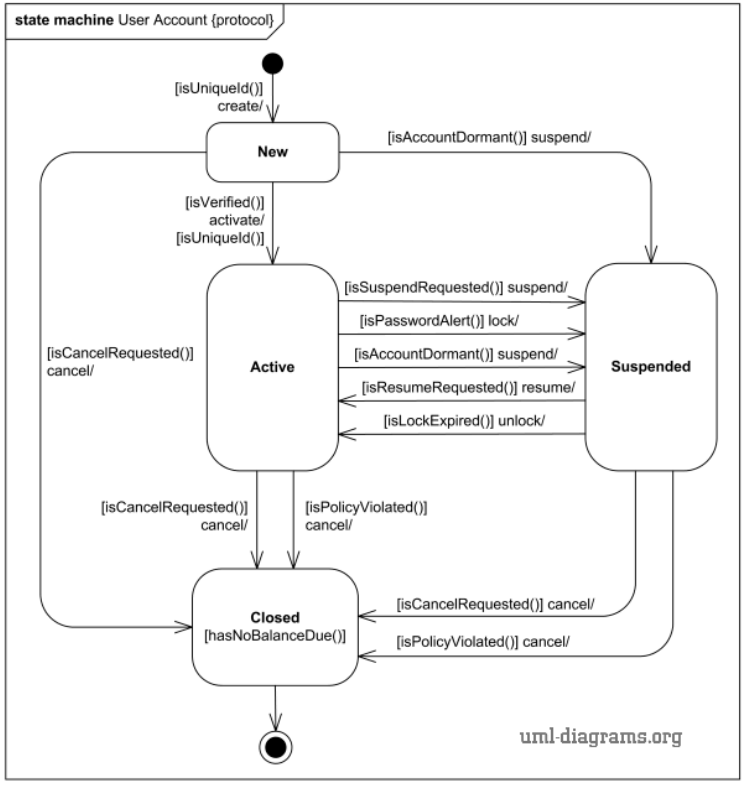
Example of MBT from a state diagram: online shopping user account



How do we identify test cases?

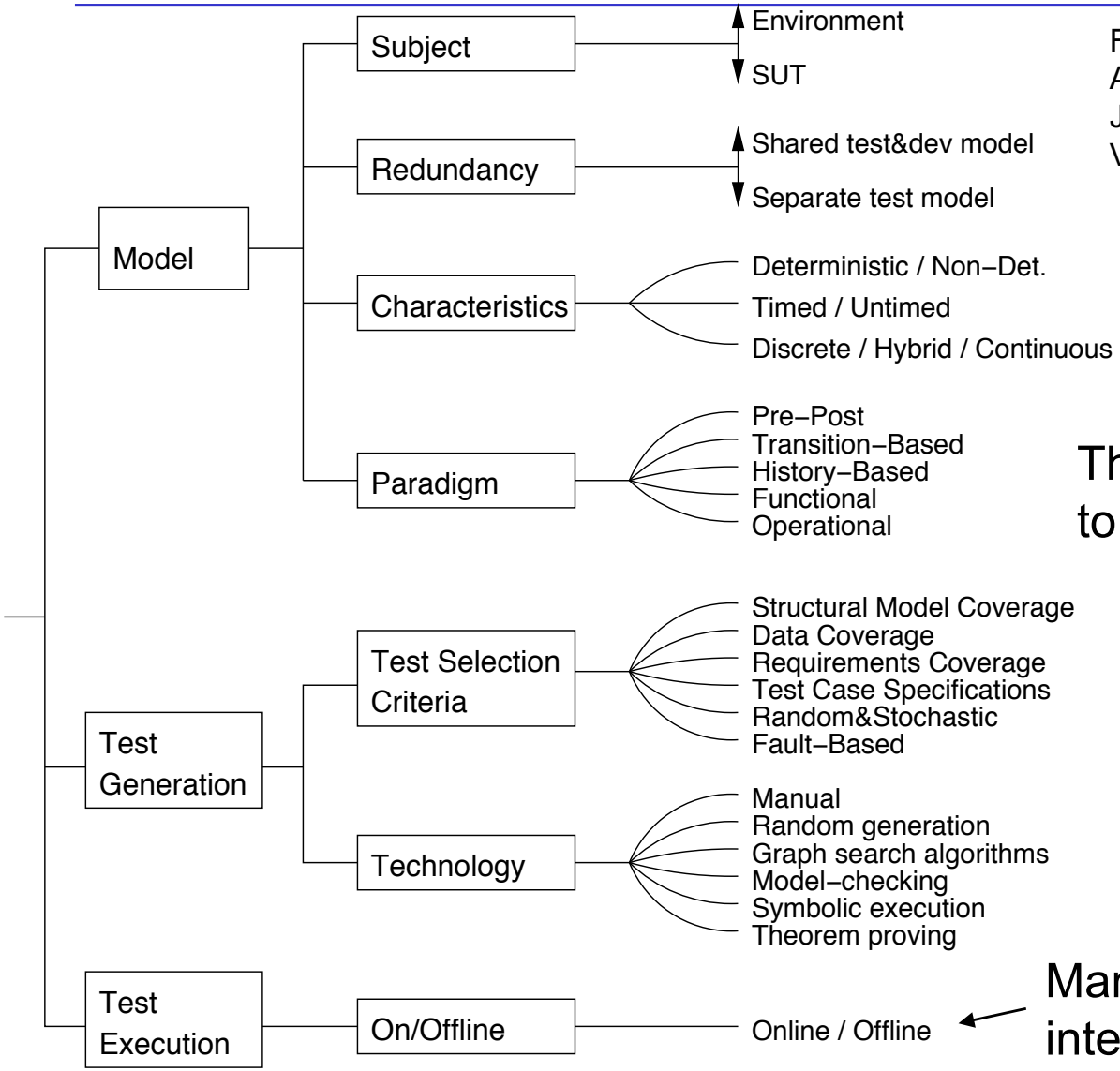
- We can “cover” the state diagram
- State coverage:
  - Every state in the model should be visited by at least one test case
- Transition coverage
  - Every transition between states should be traversed by at least one test case.
  - This is the most commonly used criterion
    - A transition can be thought of as a (precondition, postcondition) pair

Transition coverage



- create – suspend – cancel – end
- create – activate – suspend – cancel – end
- …. We continue until we consider all transitions

How to handle the phases of MBT?

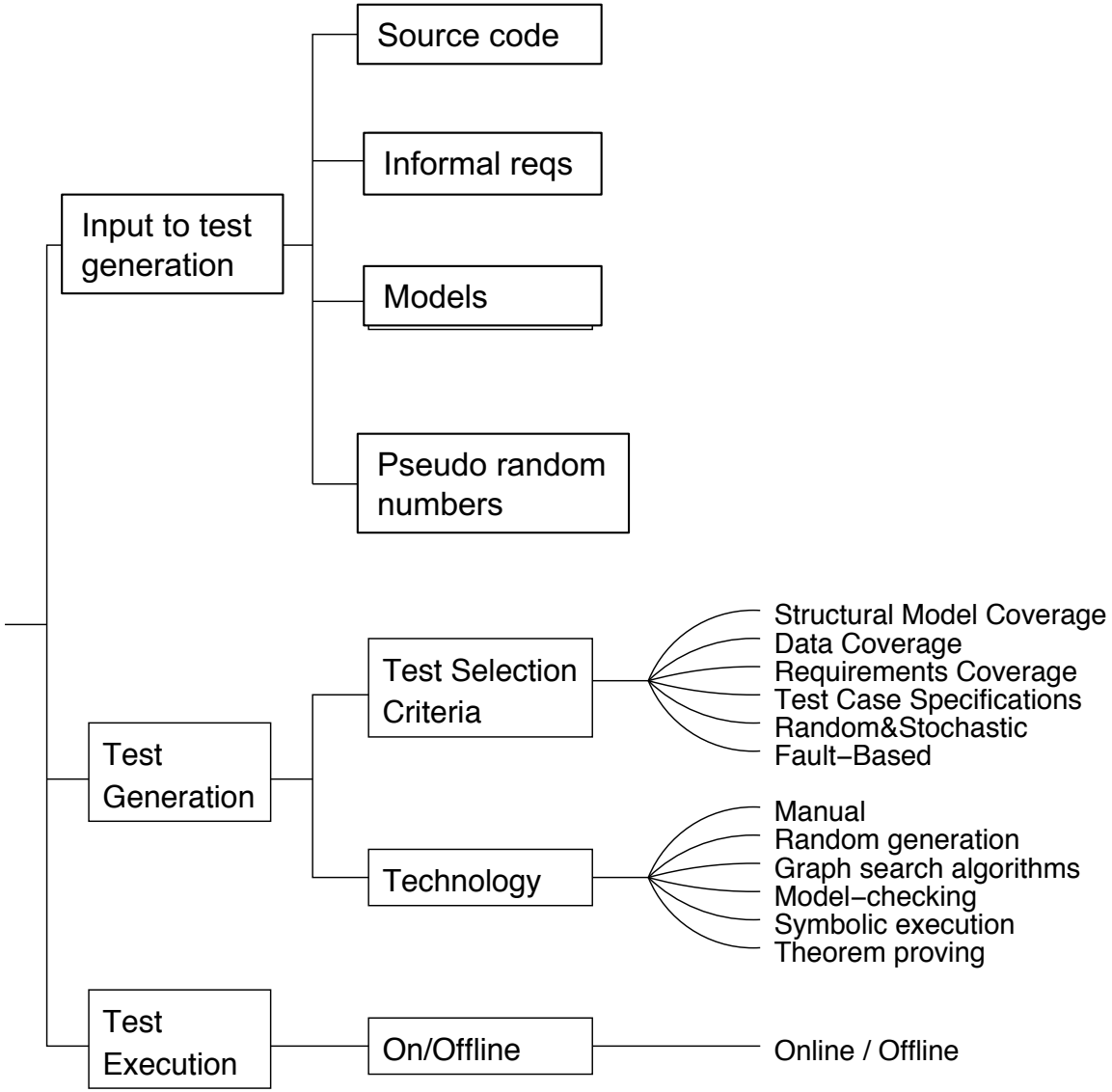


From Utting, Pretschner, Legeard,
A Taxonomy of Model-Based Testing
Journal Software Testing, Verification & Reliability
Volume 22 Issue 5, August 2012 Pages 297-312

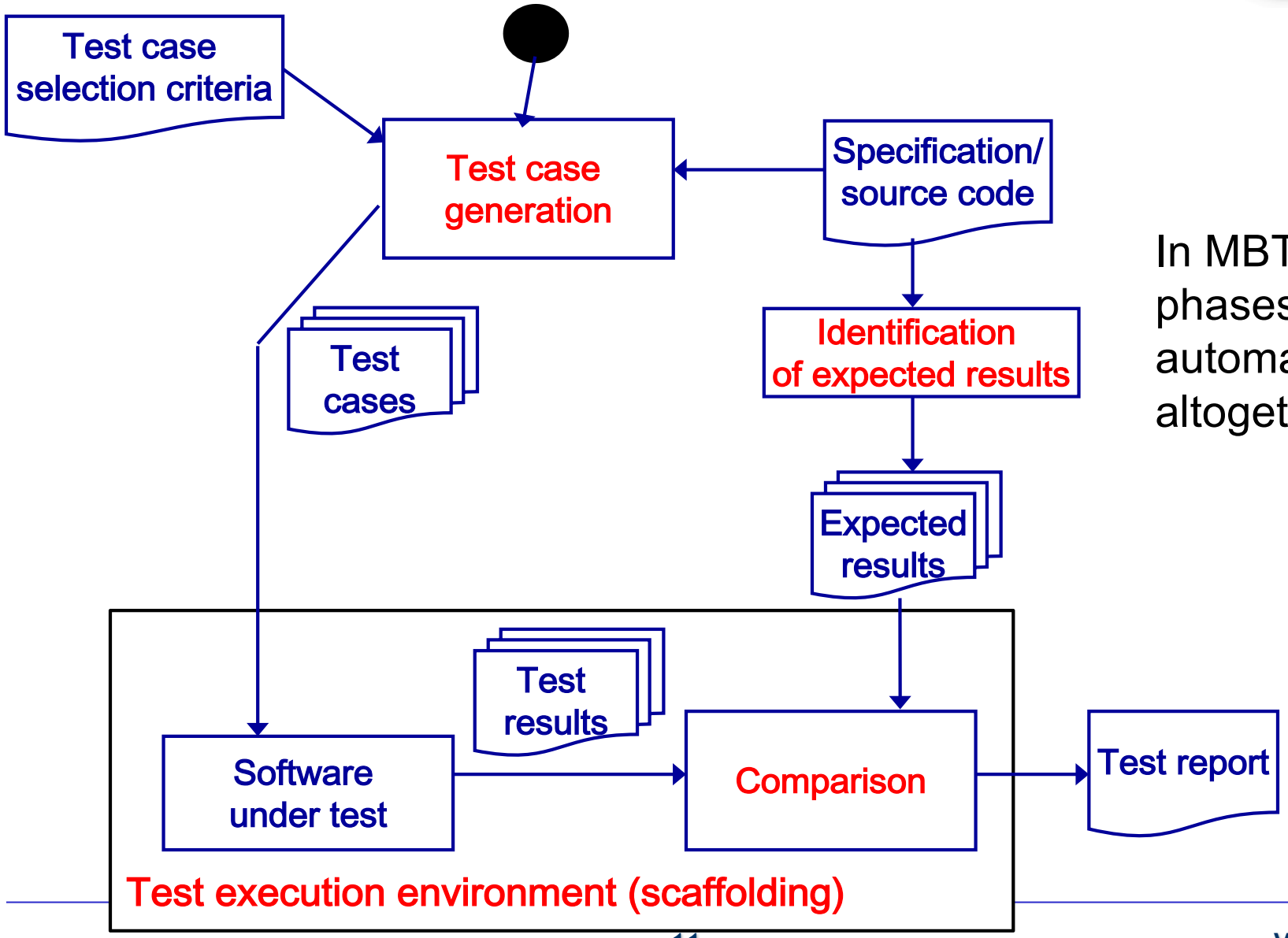
The model is the main input to the test generation

Manual vs automated and integrated with test generation

Generalizing the taxonomy …

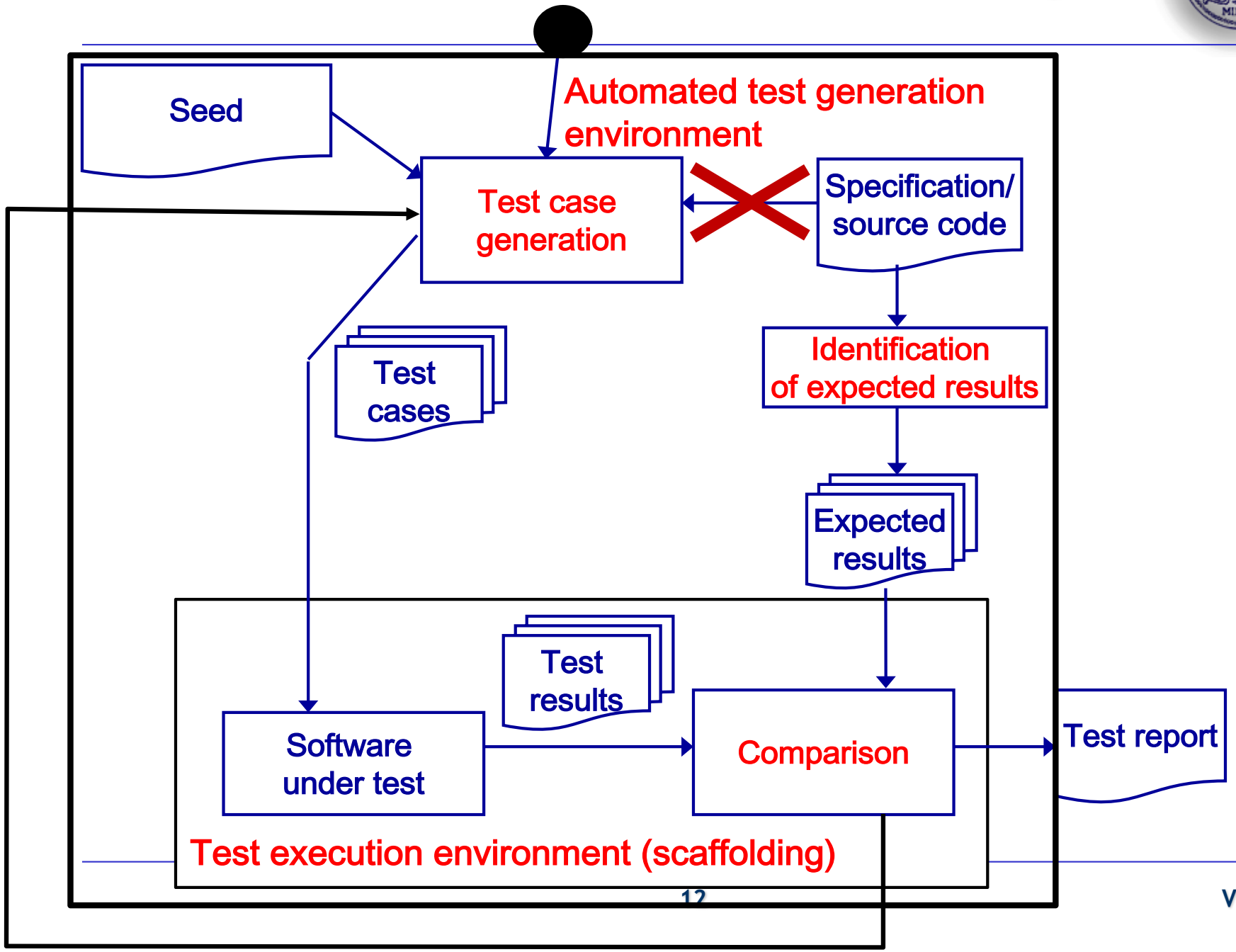


What can be automated (in red)?



In MBT these phases can be automated altogether

The case of Random/Statistical Testing



Random testing: strengths and weaknesses

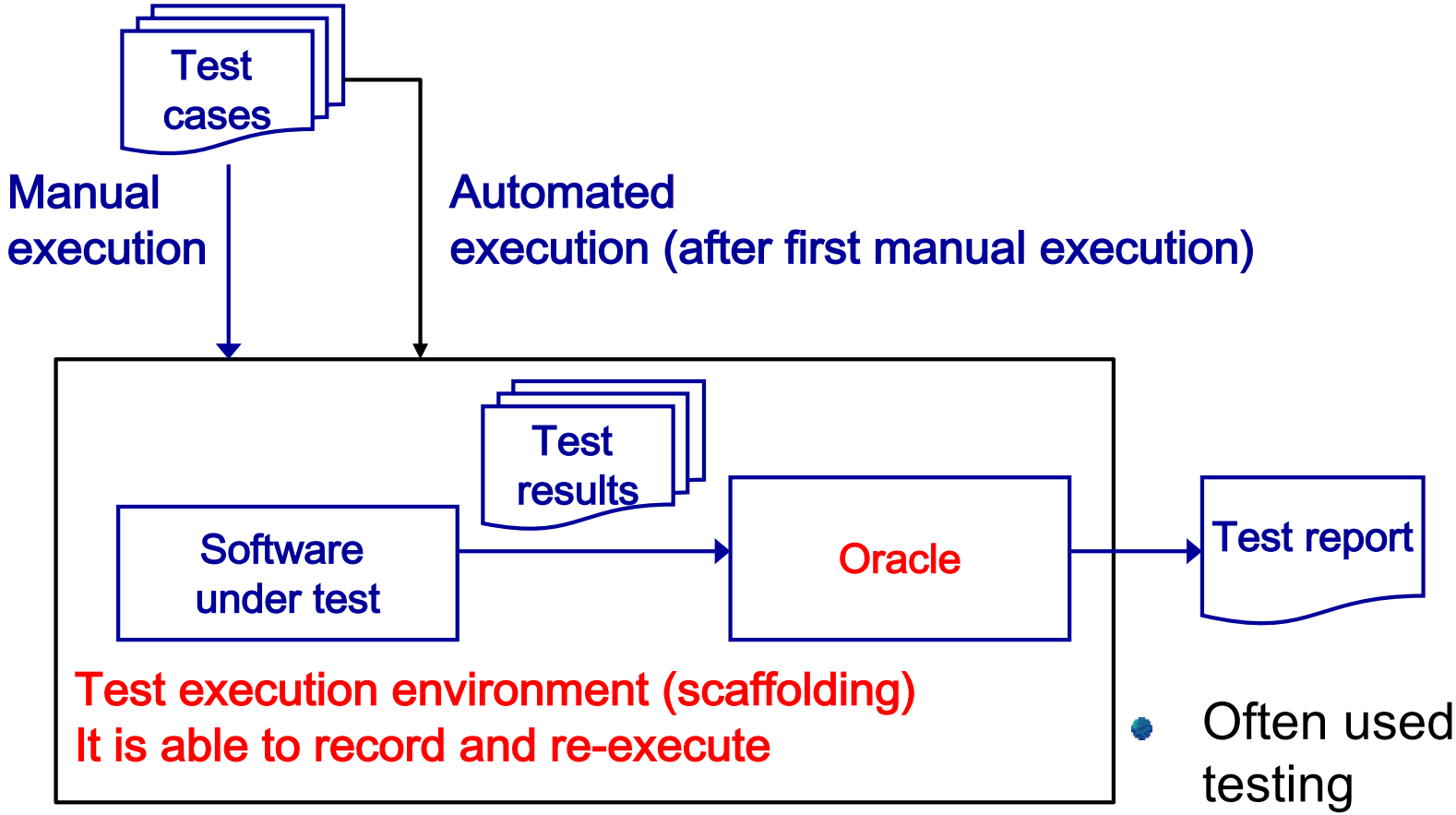
+

- Completely automatic
- Can be very extensive
- Unbiased
- Allows testers to identify completely unforeseen issues

-

- Input validity
- May find the same bug over and over
- May identify several minor bugs not relevant to most of the use cases for the software
  - May result in frustration

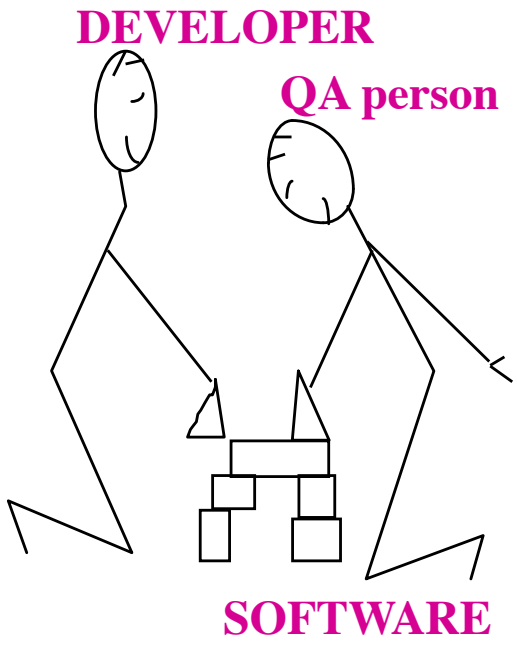
Capture & reply testing



- Often used for GUI testing
- Auto-regression testing
- First test expensive
- Low level recorded tests

Conclusion

- Verification and validation take place in most phases of development process
- Many techniques available
- A lot of possibilities for automation
- We need to avoid the attitude: yes, I'm great in my work so I do not need verification/validation
- We need to plan for the V&V process carefully
- We need to pay attention to the cohesion of the team



References

- C. Ghezzi, M. Jazayeri, D. Mandrioli Fundamentals of Software Engineering, 2nd Edition, Prentice Hall, 2003 (Italian translation Pearson Ed. 2004)
- M. Pezzè, M. Young Software Testing and Analysis: Process, Principles, and Techniques. 2008, John Wiley & Sons.
- Michael Fagan, Advances in Software Inspections, IEEE Transactions on Software Engineering, July 1986 https://doi.org/10.1109/TSE.1986.6312976
- NASA, Software Formal Inspections Guidebook, Office of Safety and Mission Assurance, NASA-GB-A302 approved August 1993, https://ntrs.nasa.gov/search.jsp?R=19980228472
- NASA, Software Formal Inspections Standard, NASA-STD-8739.9, June 2013 (supersedes NASA-GB-A302), https://standards.nasa.gov/standard/nasa/nasa-std-87399
- Check list for inspections of Java code by Christopher Fox (available on the course web site)
- A tool for supporting code review (among other things) http://phabricator.org

References

- Code reviews for teams too busy to review code (video) https://www.youtube.com/watch?v=1m3eRFeCInY
- Bacchelli, Bird, Expectations, Outcomes, and Challenges Of Modern Code Review http://research.microsoft.com/pubs/180283/ICSE%202013-codereview.pdf