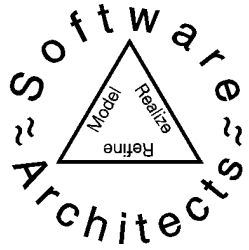
Guided Inspection

Melissa L. Major
John D. McGregor



Software Architects
P.O. Box 3104
Collegedale, TN 37315

Phone:423.238.3288
Fax: 423.238.3289

www.software-architects.com
info@software-architects.com

SEW '99

Outline

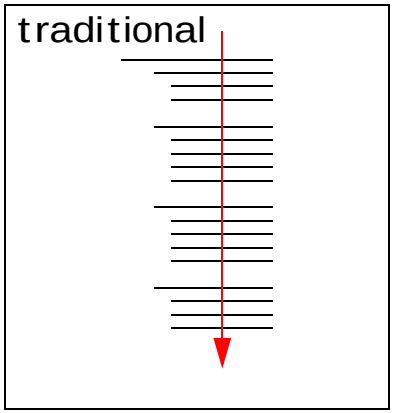
- Basics and Definitions
- Selecting Test Cases
- Application of Technique
- Evaluation of Technique

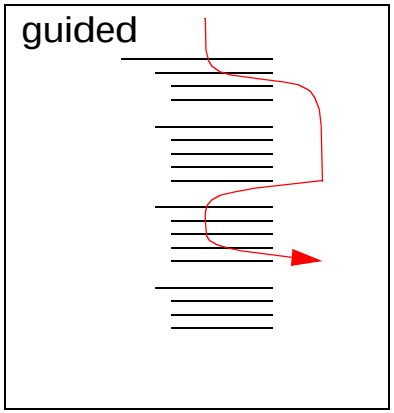
Problem

- Existing inspection/review techniques examine ***what is in the model*** for errors
- There is no systematic way to consider ***what should be in the model***
- ***Guided Inspection*** is a technique that supplements rigorous inspection/review techniques, that address model syntax, with test cases to systematically examine the semantics of the models

What's Different?

- Guided Inspection does not move sequentially
- What is inspected next depends upon the scenario/semantics
- The inspection can be driven by customer priorities.
- The inspection can be focused to identify specific types of defects



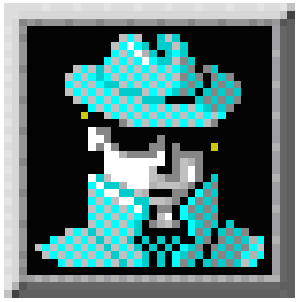


Guided Inspection Outline

- Analyze the model to be inspected
- Complete checklist for appropriate model
- Systematically sample to select test cases
- Write down the test cases
- Apply to the model to be inspected
- Analyze the model to determine coverage levels

Roles

- Tester
  - Select and write test cases
- Developer
  - Perform symbolic execution
- Manager
  - Stay out of the way - this is fault finding - conduct your own evaluation separately

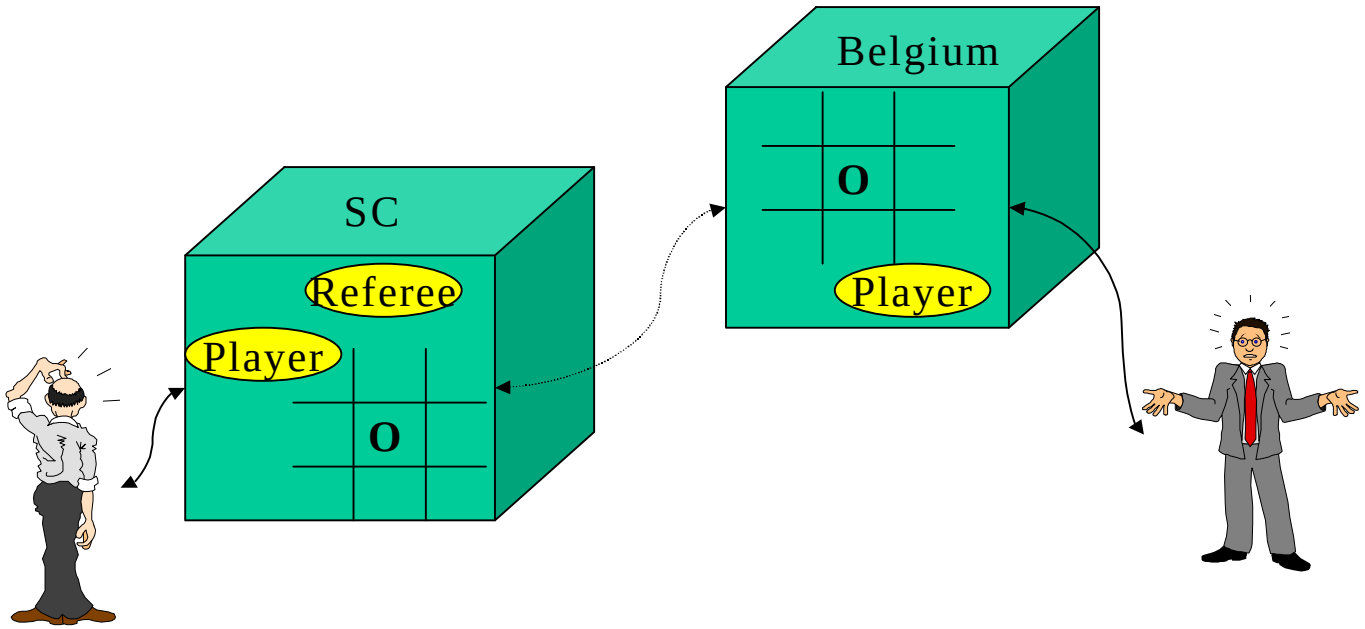


Evaluation Criteria for Models

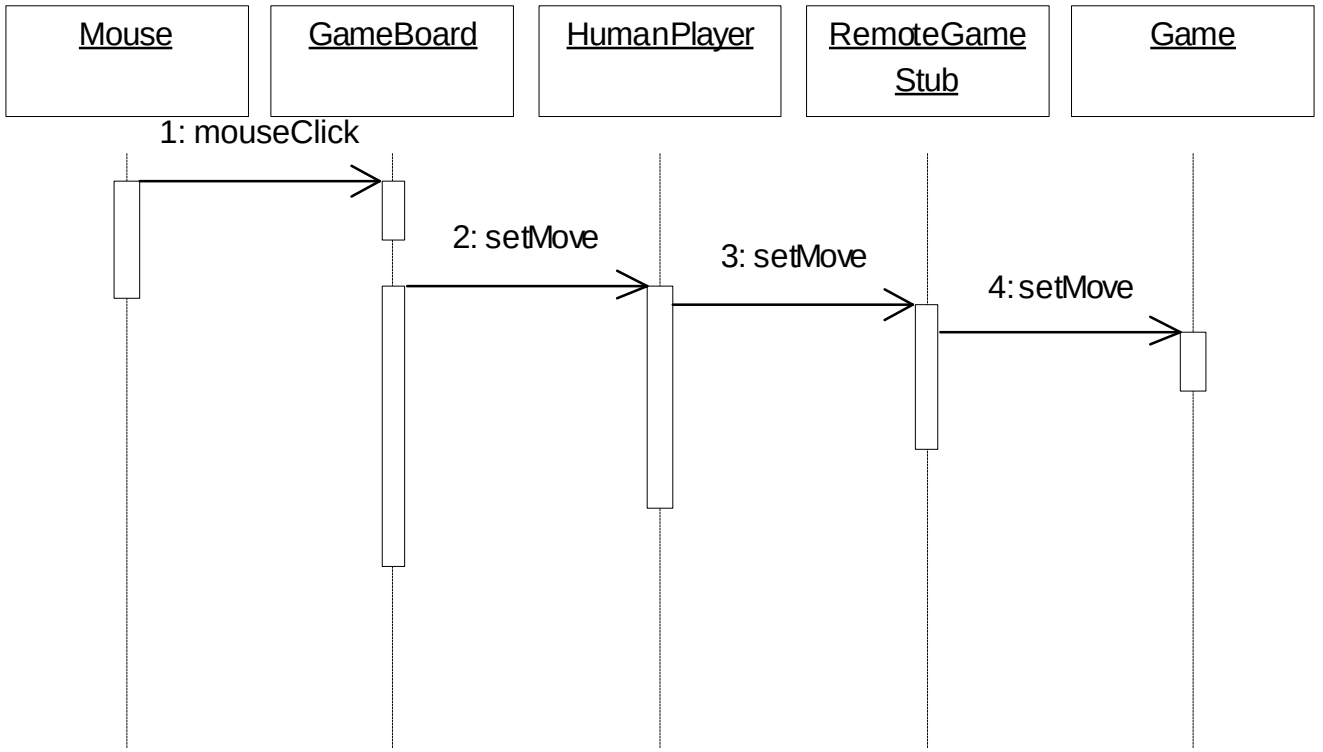
- Completeness
  - Are there scenarios the model can not handle?
- Correctness
  - Does the model handle each scenario accurately?
- Consistency
  - Does one part of the model contradict other parts of the model?

Case Study

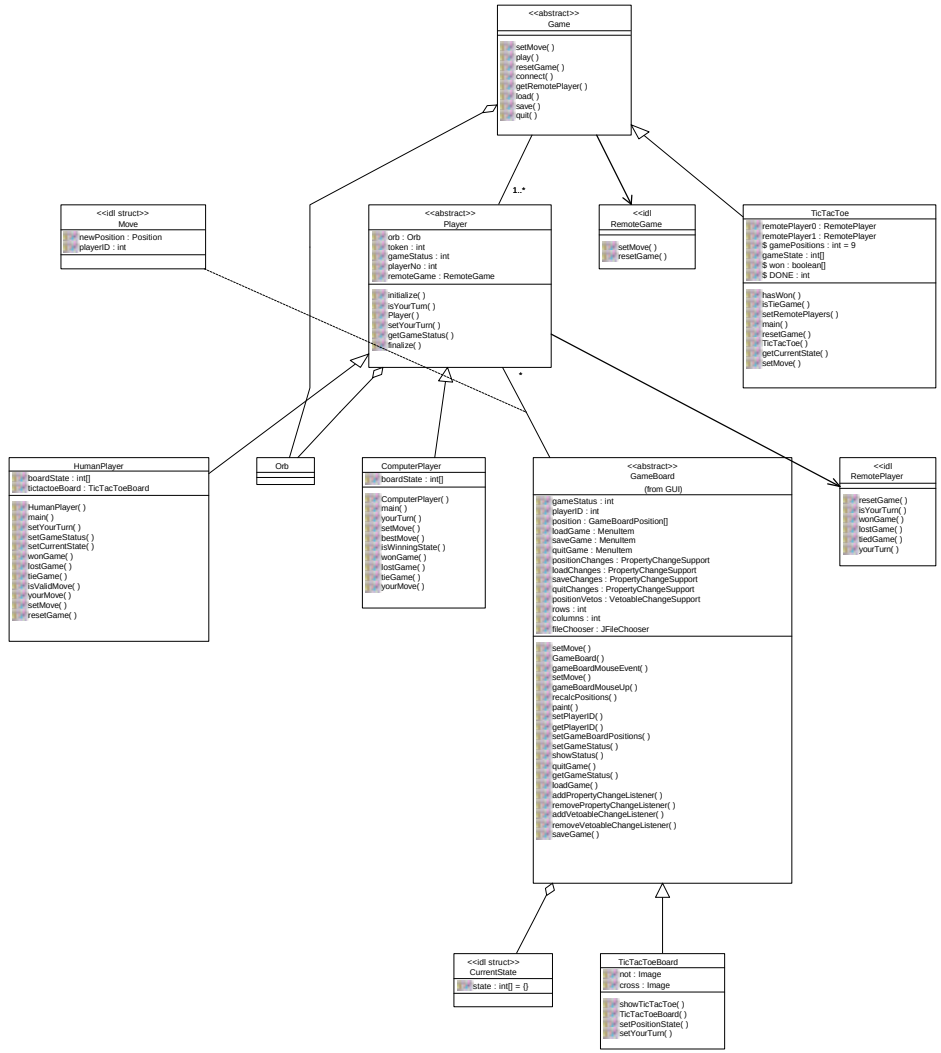
- TicTacToe - a distributed, component-based implementation
- Guided Inspection assumes a completed UML model



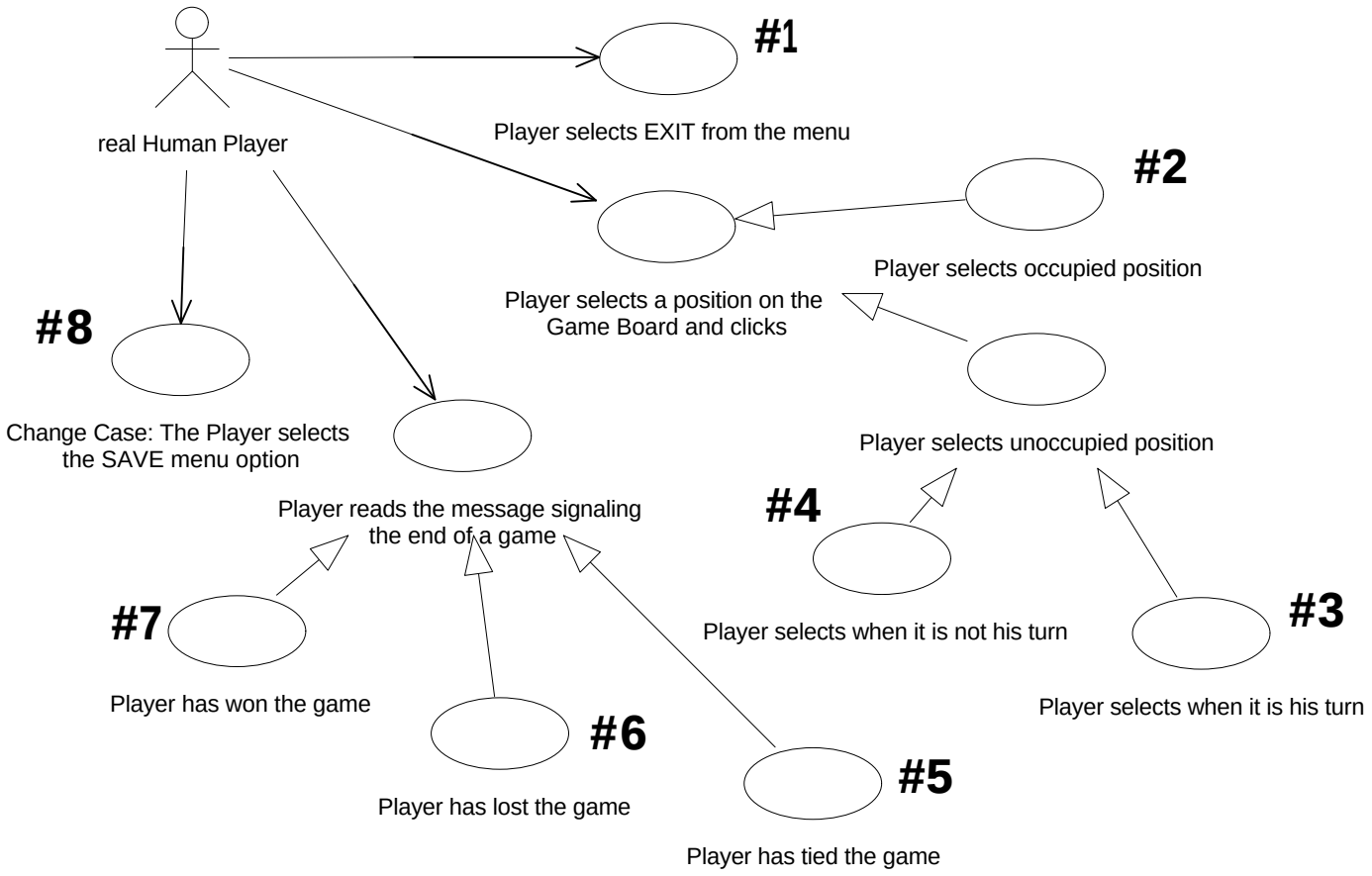
Sequence Diagram for setMove



Design-level Class Diagram



Use Cases


real Human Player
#1
Player selects EXIT from the menu
#2
Player selects occupied position
Player selects a position on the Game Board and clicks
#8
Change Case: The Player selects the SAVE menu option
Player selects unoccupied position
Player reads the message signaling the end of a game
#4
Player selects when it is not his turn
#3
Player selects when it is his turn
#7
Player has won the game
#6
Player has lost the game
#5
Player has tied the game

CheckList

- Addresses the syntax of the model
- May be replaced with automated syntax checking

Domain Analysis Model Checklist

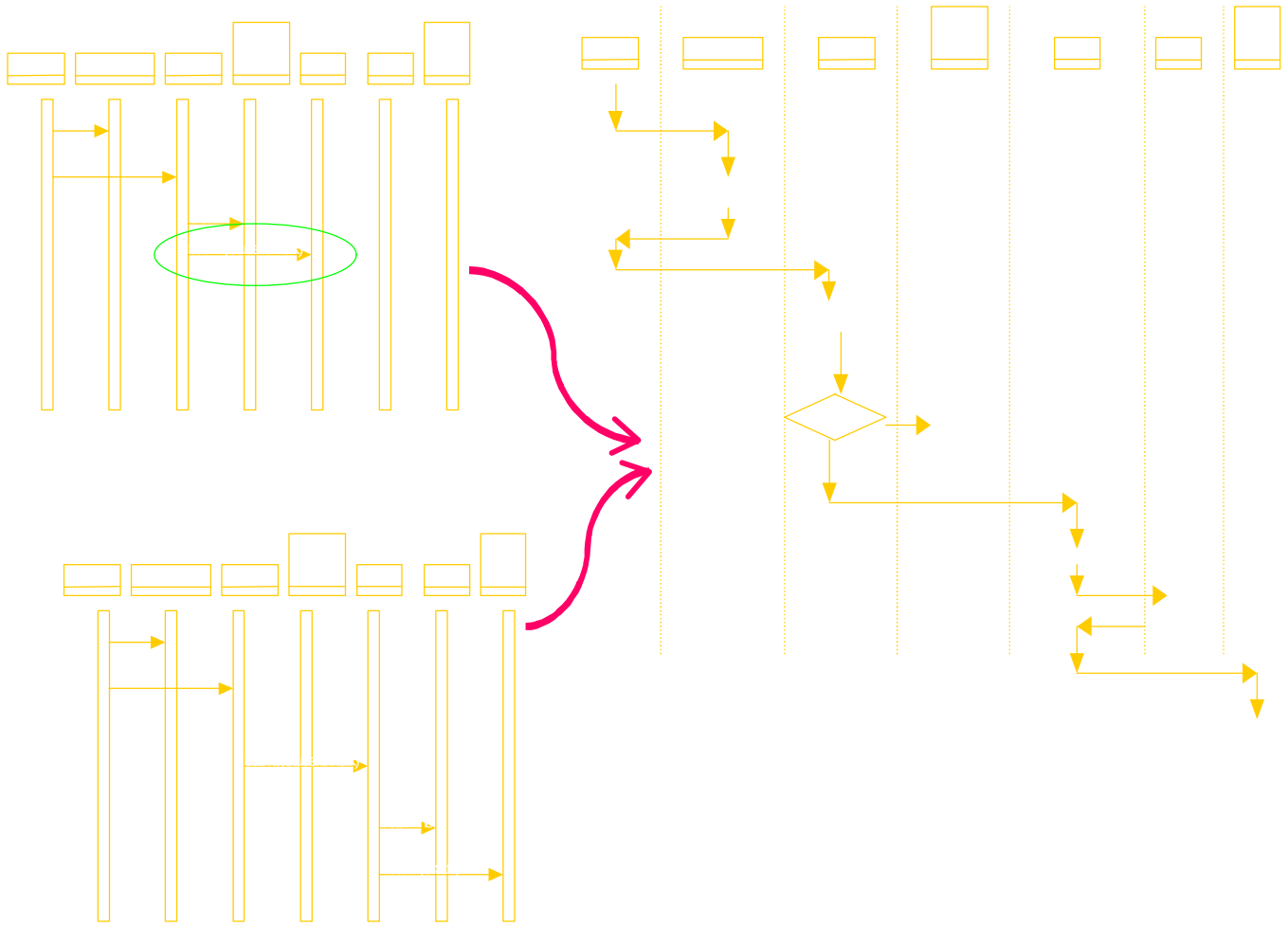
Each class box is connected to at least one other class box. _____ Yes _____ No

Each className is a recognized term in the domain. _____ Yes _____ No

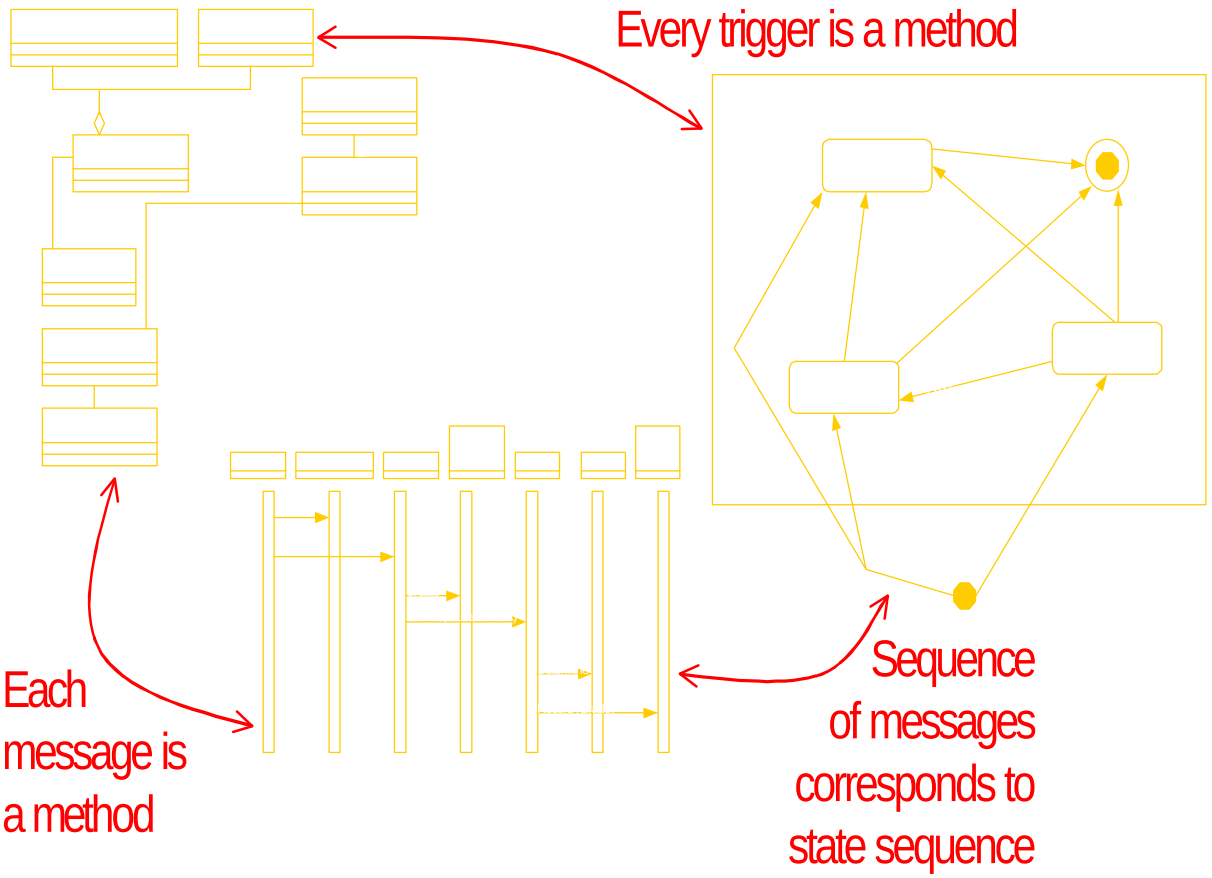
Each subclass of a class is a specialization of the original class. _____ Yes _____No

....

Consistency between diagrams



Consistency between diagrams



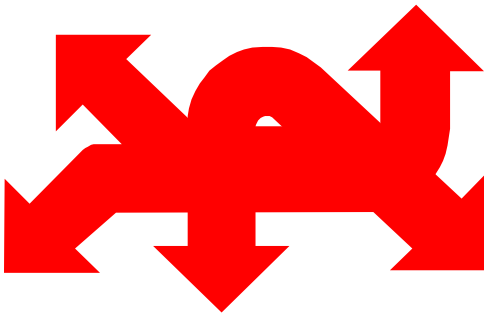
Building System Test Cases

- Select a scenario
  - The Player selects an unoccupied slot when it is his turn. The system responds by displaying the Player's icon in the appropriate position on the GameBoard.
- Write test cases by assigning each variable a specific value.
  - The Player selects the center slot of an empty board, **OR**
  - The Player selects the slot in location (1,1) when slots (0,1) and (2,1) are taken by the opponent and Player has (2,2).

Prioritizing

- Each use case is annotated with three attributes:
  - frequency - how often will this feature be used relative to other features of the program?
  - criticality - how necessary is this feature to the success of the product?
  - risk - how likely is there to be a problem in implementing this feature?
- Each attribute is valued on a scale from **Low** to **High**.

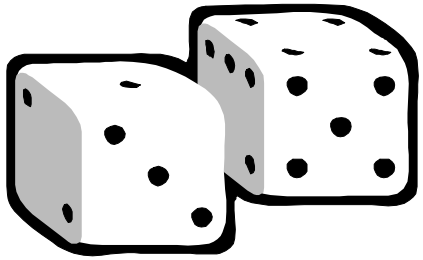
Combining Attribute Values



- For a single use case, we have three attribute values.
- Risk is used for scheduling development increments
- Frequency and criticality are both useful for testing:
  - the most often used, most necessary feature should be tested the most
- If criticality is **HIGH** and frequency is **LOW**
  - **conservative** combined value - **HIGH**
  - **averaging** combined value - **MEDIUM**

Sampling

- Use cases that have a combined frequency/ criticality rating of high will be tested over a wider range than those with a low rating.
- Equivalence classes are established for each variable



- Test cases are formed by selecting values from the equivalence classes
- A value for a field is chosen and paired with values of each equivalence class for each variable

Rationing Test Cases

- The number of test cases will vary based on the priority of the use case.

Use Case #	Frequency	Criticality	Risk	Combined Frequency/ Criticality	Number of test cases
1	Low	Low	Low	Low	1
2	Low	Medium	High	Medium	3
3	High	High	Medium	High	9
4	Low	High	Medium	Medium	3
5	Low	Medium	Medium	Medium	3
6	Low	Medium	Medium	Medium	3
7	Low	Medium	Medium	Medium	3

- Start with the extremes. There are only 9 test cases for use case #3 and 1 for use case #1. This covers these possibilities exhaustively.
- Select a number inbetween for the medium ones. 3 may be a bit low but there may be insufficient time to do more.

Inspection Session

- Testers guide the inspection by setting the scenario
- Developers “describe” the execution using their knowledge of the classes, but also referring to pre and post-conditions.
- Developers record the execution using an appropriate UML diagram.

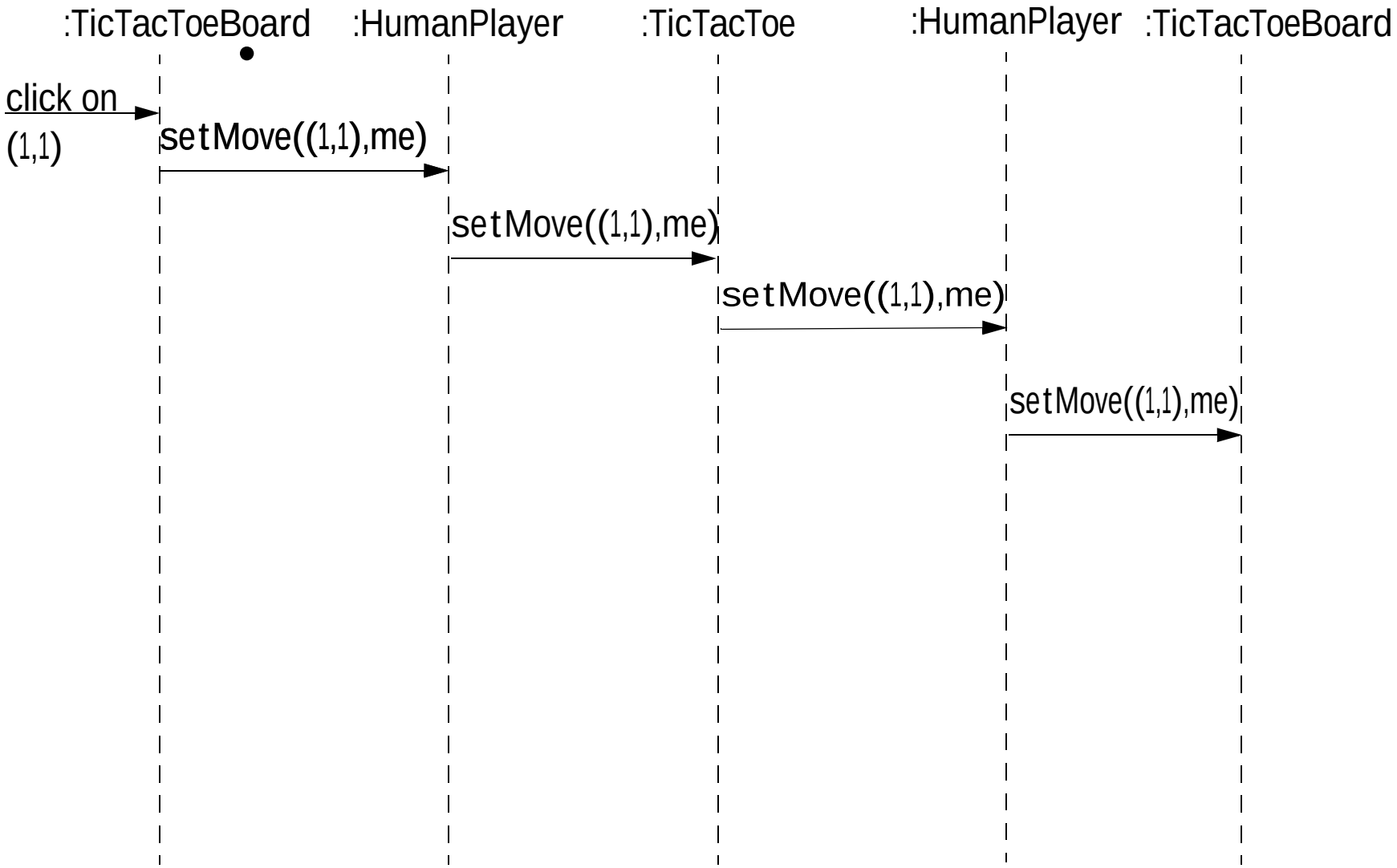
Symbolic Execution

- The tester selects a test case, using the sampling technique.
- The tester defines the state of the system at the start of the test case. Label the state of each object at the start of the test as state 0.
- The tester initiates the trigger of the use. The developer identifies the message sent and records it on the sequence diagram. If an object changes state bcause of the message, label the new state, state1.
- Repeat until the scenario is completed.

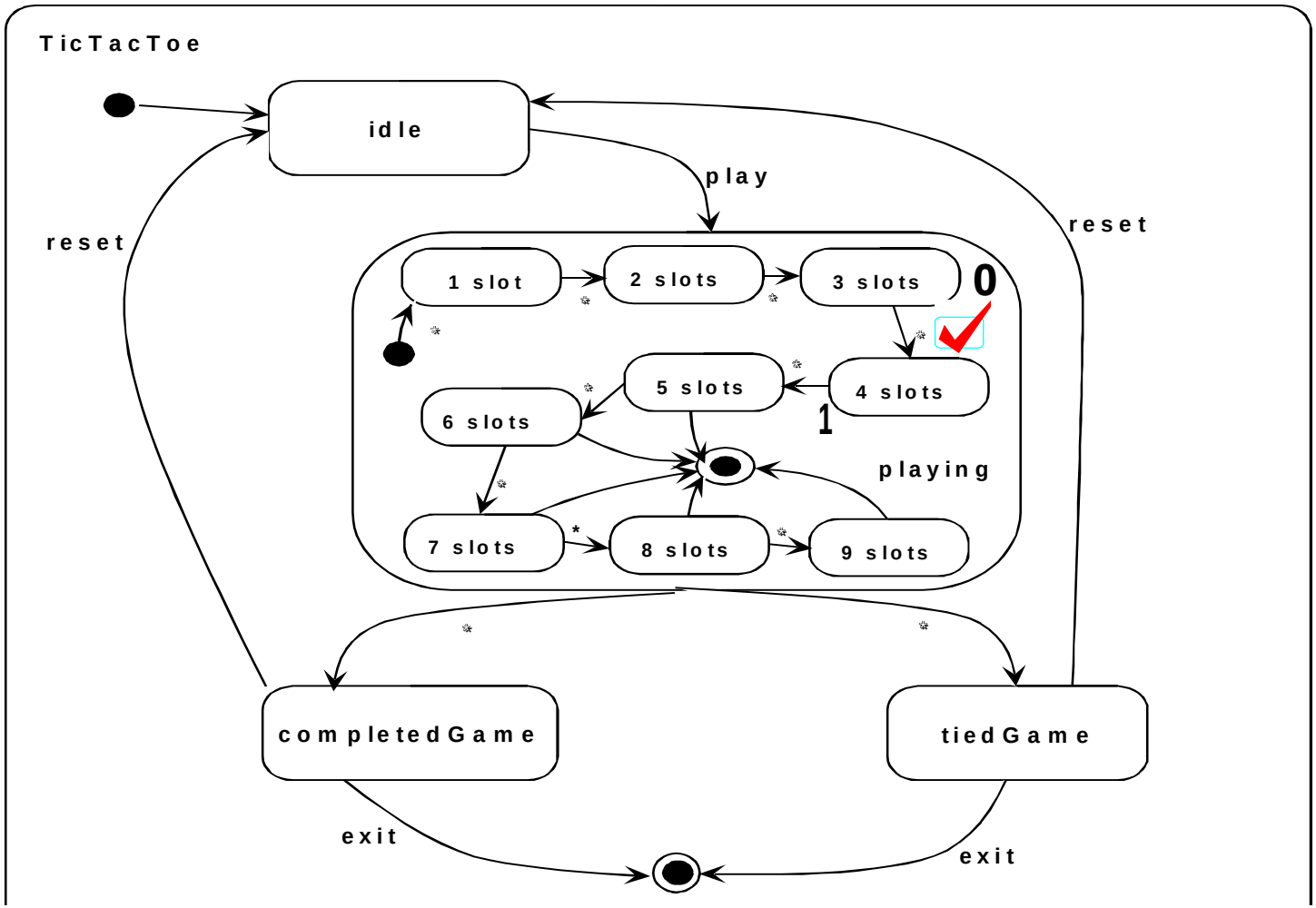
Example Test Case

- Test scenario: The Player selects the slot in location (1,1) when slots (0,1) and (2,1) are taken by the opponent and Player has (2,2).
- Expected result: The board is refreshed and shows the appropriate icon in the selected cell.
- Several steps are needed as pre-conditions for this test case
- These steps can be parts of other test cases.
- Give attention to sequencing tests to minimize the effort required for each test.

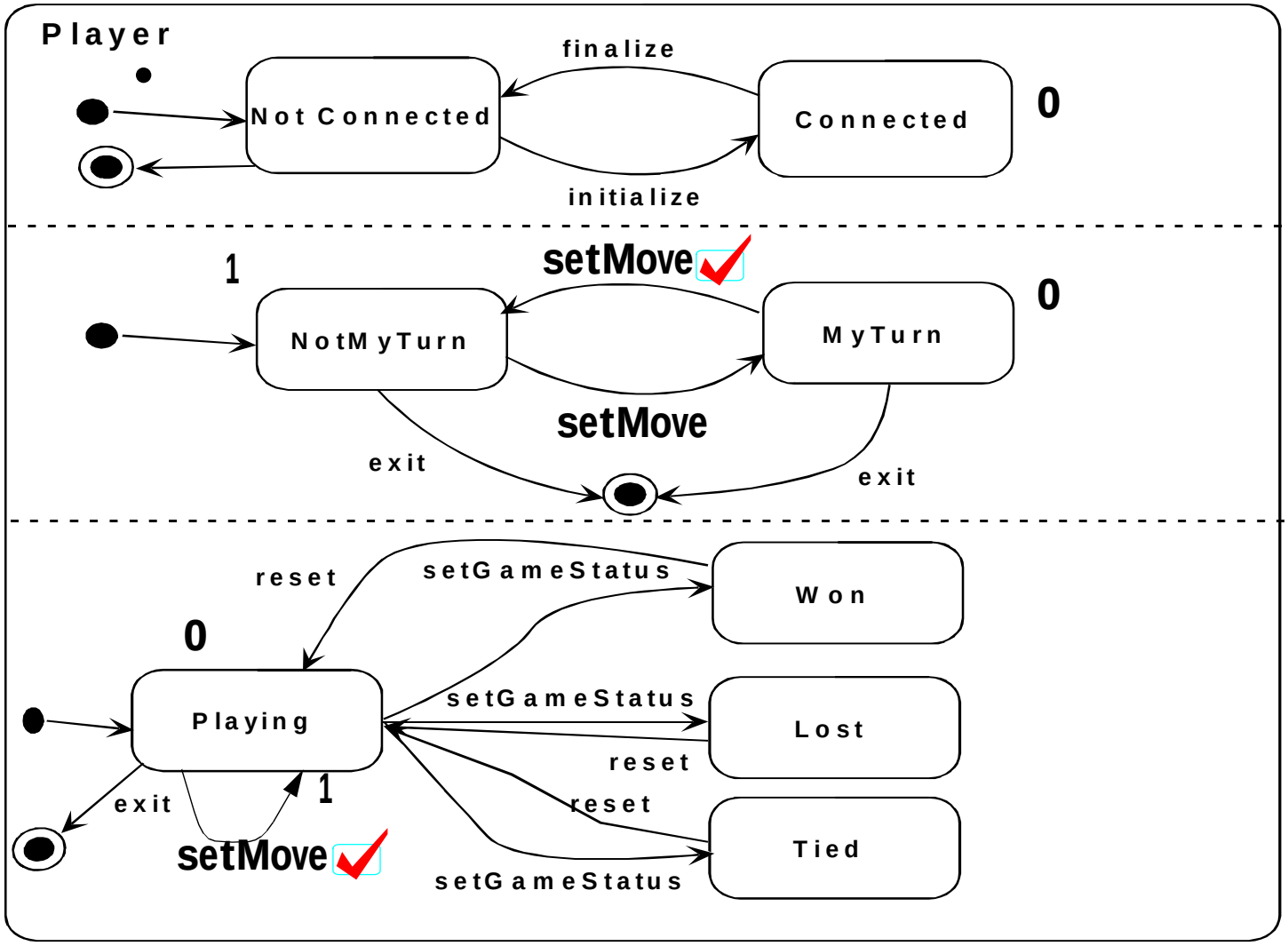
Example Test Case



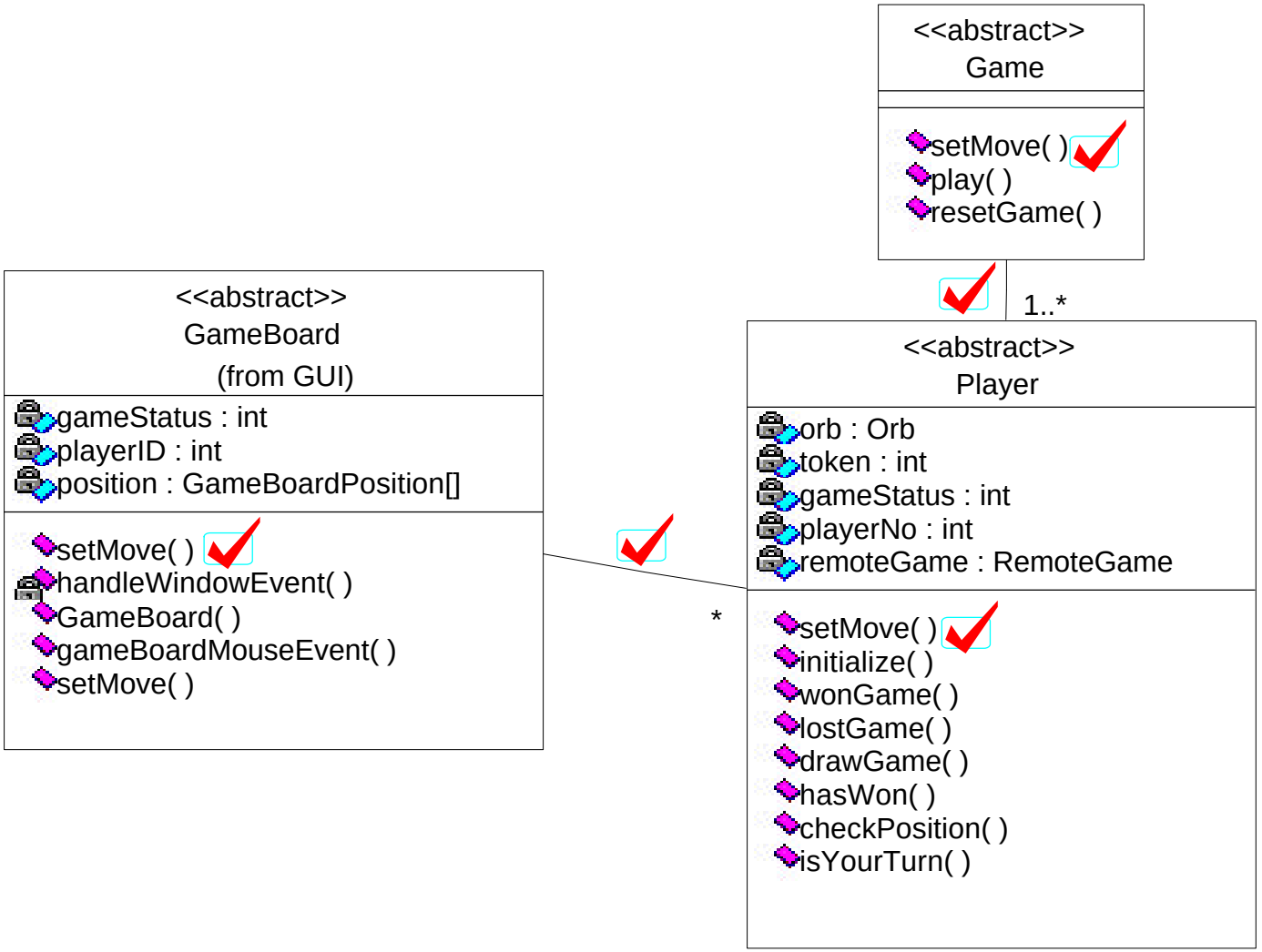
Example Test Case



Example Test Case

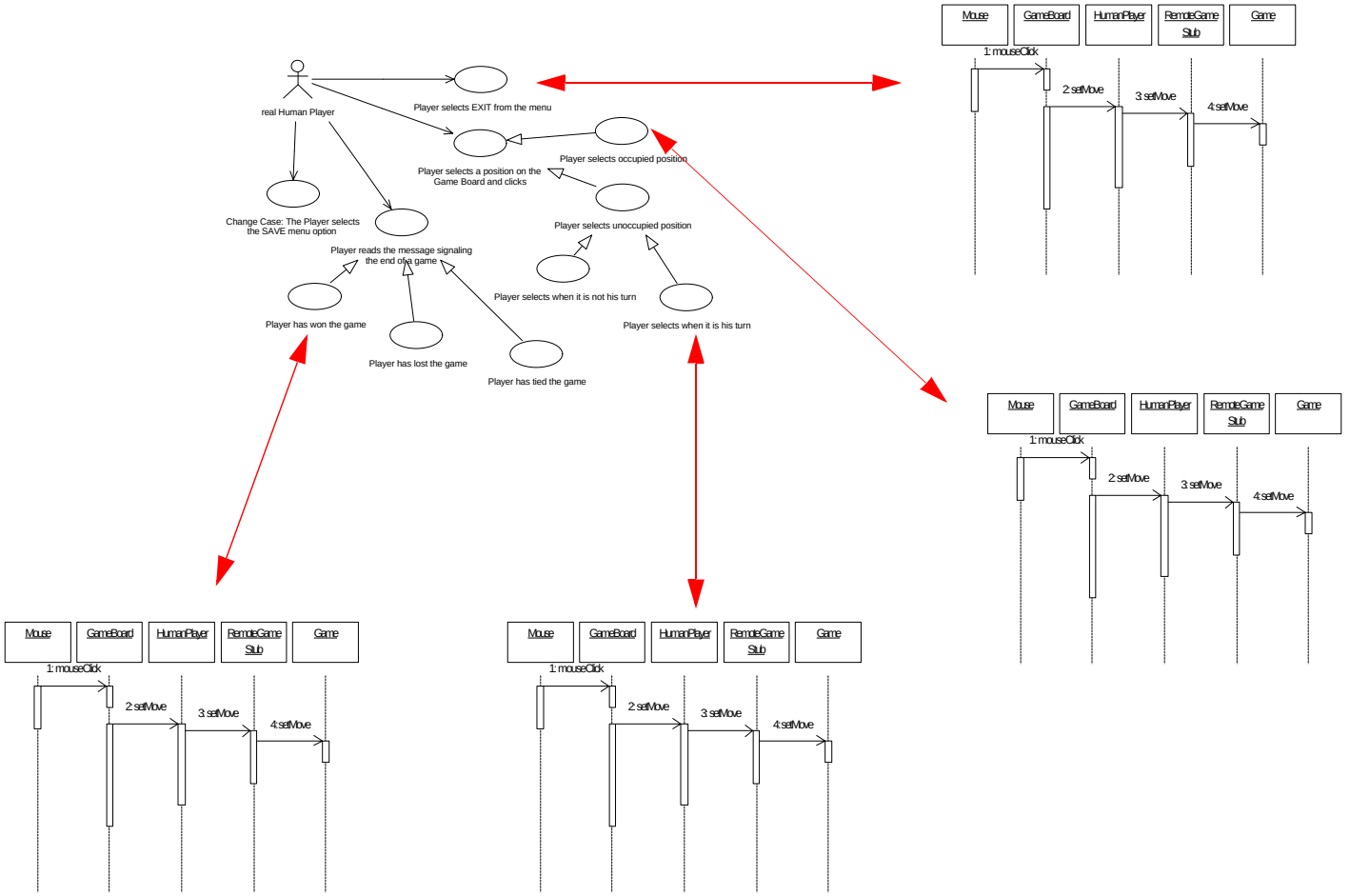


Example Test Case



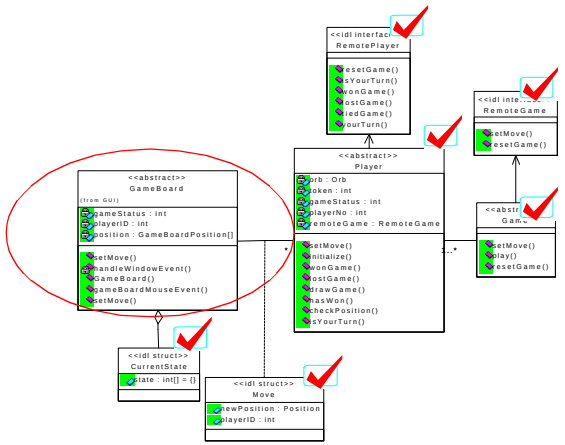
Coverage Analysis

- Is there at least one scenario for each use case?



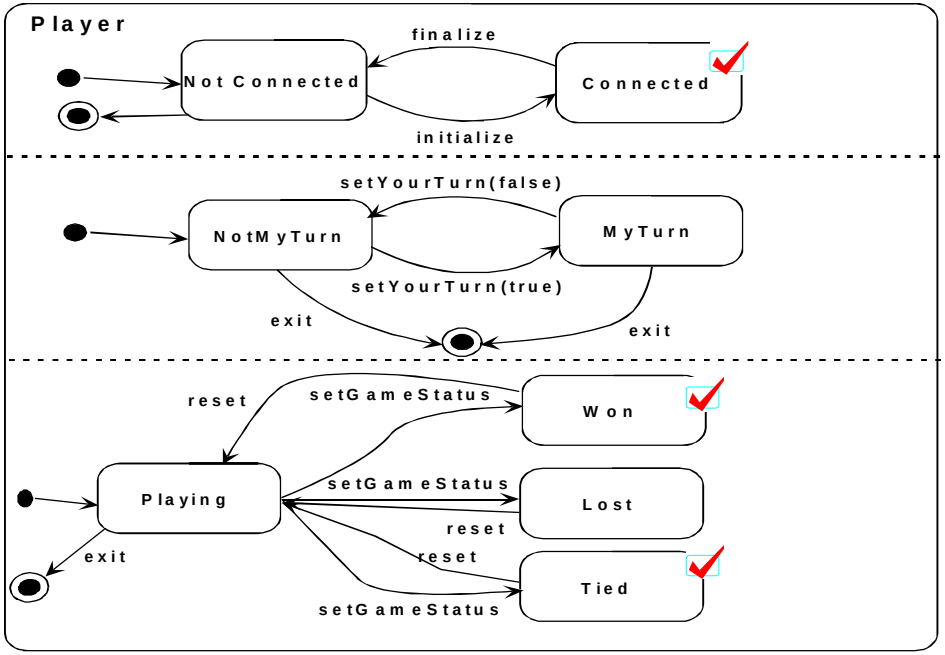
Coverage Analysis

- Has every class been touched by a test case? If not why not?
- Are there important methods that have not been used?



Coverage Analysis

- Has every important state of every important class been used in a scenario?
- If not, what scenario would reach this state?



Conclusion

- The system has been analyzed from three perspectives: correctness, completeness and consistency.
- Companies have reported that it costs as much as 100 times more to repair a defect at system test time as it would to repair at analysis time
- While guided inspection is a person intensive technique, even the early expenditure of considerable resources can still result in a net savings over the full project life cycle

Thanks

- On behalf of Software Architects, thank you for attending this session.
- Please remember us for your future object oriented training and consulting needs.
- My e-mail address is:

major@software-architects.com

Please keep in touch if there is anything I can do for you.

A Process Definition for Guided Inspection

John D. McGregor

Melissa L. Major

Software Architects

Goal: To identify defects in artifacts created during the analysis and design phases of software construction.

Steps in the Process

Define the scope of the GI

Identify the basis model(s) from which the material being inspected was created

Assemble the GI team

Define a sampling plan and coverage criteria

Create test cases from the bases

Apply the checklist and tests to the material

Gather and analyze test results

Report and feedback

Detailed Step Descriptions

Define the scope of the review/inspection

Inputs:

the project's position in the life cycle

the materials produced by the project (UML models, plans, use cases)

Outputs:

A specific set of diagrams and documents that will be the basis for the evaluation

Method:

Define the scope of the GI to be the set of deliverables from a phase of the development process. Use the development process information to identify the deliverables that will be produced by the phase of interest.

Example:

The project has just completed the domain analysis phase. The development process defines the deliverable from this phase as a UML model containing domain-level use cases, static information such as class diagrams and dynamic information such as sequence and state diagrams. The GI will evaluate this model.

Identify the basis model(s) from which the material being inspected was created

Inputs:

The scope of the GI

The project's position in the life cycle

Outputs:

The material from which the test cases will be constructed (The Model Under Test - MUT)

Method:

Review the development process description to determine the inputs to the current phase. The basis model(s) should be listed as inputs to the current phase.

Example:

The inputs to the domain analysis phase is the “knowledge of experts familiar with the domain”. These mental models are the basis models for this GI.

Assemble the GI team

Inputs:

The scope of the GI

Available personnel

Outputs:

A set of participants and their roles

Method:

Assign persons to fill one of three categories of roles: Administrative, Participant in creating the model to be tested, Objective observer of the model to be tested. Choose the objective observers from the customers of the model to be tested and the participants in the creation of the basis model.

Example:

Since the model to be tested is a domain analysis model and the basis model is the mental models of the domain experts, the objective observers can be selected from other domain experts and/or from application analysts. The creation participants are members of the domain modeling team. The administrative personnel can perhaps come from other interested parties or an office that provides support for the conduct of GIs.

Define a sampling plan and coverage criteria

Inputs:

The project's quality plan

Outputs:

A plan for how test cases will be selected

A description of what parts of the MUT will be covered

Method:

Identify important elements of this MUT. Estimate the effort required to involve all of these in the GI. If there are too many to cover, use information such as the RISK section of the use cases or the judgement of experts to prioritize the elements.

Example:

In a domain model there are static and dynamic models as well as use cases. At least one test case should be created for each use case. There should be sufficient test cases to take every “major” entity through all of its visible states.

Create test cases from the bases

Inputs:

The sampling plan

MUT

Outputs:

A set of test cases

Method:

Obtain a scenario from the basis model. Determine the pre-conditions and inputs that are required to place the system in the correct state and to begin the test. Present the scenario to the “oracle” to determine the results expected from the test scenario. Complete a test case description for each test ase.

Example:

A different domain expert than the one who supported the model creation would be asked to supply scenarios that correspond to uses of the system. The experts also provide what they would consider an acceptable response.

Apply the checklist and tests to the material

Inputs:

Checklist for the type of model being inspected

Set of test cases

MUT

Outputs:

Set of test results

Completed checklist

Method:

Apply the test cases to the MUT using the most specific technique available. For UML models in a static environment, such as Rational Rose, an interactive simulation session in which the Creators play the roles of the model elements is the best approach. If the MUT is represented by an executable prototype then the test cases are mapped onto this system and executed. After the model has been thoroughly examined, complete the checklist.

Example:

The domain analysis model is a static UML model. A simulation session is conducted with the Observers feeding test cases to the Creators. The Creators provide details of how the test scenario would be processed through the model. Sequence diagrams are used to document the execution of each test case. Use agreed upon symbols or colors to mark each element that is touched by a test case.

Gather and analyze test results & coverage

Inputs:

Test results in the form of sequence diagrams and pass/fail decisions

The marked-up model

Outputs:

Statistics on percentage pass/fail

Categorization of the results

Defect catalogs and defect reports

A judgement of the quality of the MUT and the tests

Method:

Begin by counting the number of test cases that passed and how many have failed. Compare this ratio to other GIs that have been conducted in the organization. Compute the percentage of each type of element that has been used in executing the test cases. Use the marked-up model as the source of this data. Update the defect inventory with information about the failures from this test session.

Categorize the failed test cases. This can often be combined with the previous two tasks by marking paper copies of the model. Follow the sequence diagram for each failed test case and mark each message, class and attribute touched by a failed test case.

Example:

For the domain analysis model we should be able to report that every use case was the source of at least one test case, that every class in the class diagram was used at least once. Typically on the first pass, some significant states will be missed. This should

be noted in the coverage analysis.

Report and feedback

Inputs:

Test results

Coverage information

Outputs:

Information on what new tests should be created

Test report

Method:

Follow the standard format for a test report in your organization to document the test results and the analyses of those results. If the stated coverage goals are met then the process is complete. If not, use that report to return to step 5 and proceed through the steps to improve the coverage level.

Example:

For the domain analysis tests, some elements were found to be missing from the model. The failing tests might be executed again after the model has been modified.

Roles in the Process

Administrator

The administrative tasks include running the GI sessions, collecting and disseminating the results, and aggregating metrics to measure the quality of the review. In our example, the administrative work could be done by personnel from a central office.

Creator

The persons who created the MUT. Depending upon the form that the model takes, these people may "execute" the symbolic model on the test cases or they may assist in translating the test cases into a form that can be executed with whatever representation of the model is available. In our example the modelers who created the domain model would be the "creators".

Observer

Persons in this role create the test cases that are used in the GI. In our example they would be domain experts and preferably experts who were not the source of the

information used to create the model initially.