

Testing and EVO: Evolutionary Project Management

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
5.2.47
Information Systems
March 6, 2008

Goal

Agile Testing Principles

Tutorial on Design and XP reflections (d401a, s601d)

EVO

What you should learn

To apply test driven practices and to motivate them

To reflect on XP and design practices

To apply EVO

Goal

Agile Testing Principles

- Testing overview
- Unit testing environment
- Large agile testing case
- Case results
- Easy accept

Tutorial on Design and XP reflections (d401a, s601d)
EVO

Software Testing

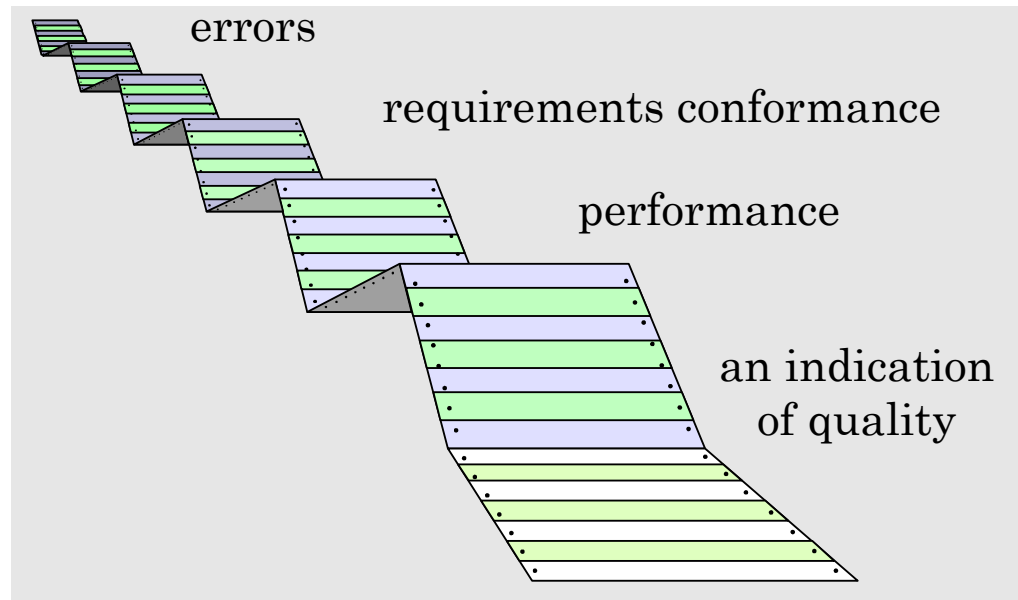
Modelling the software environment

Selecting test scenarios

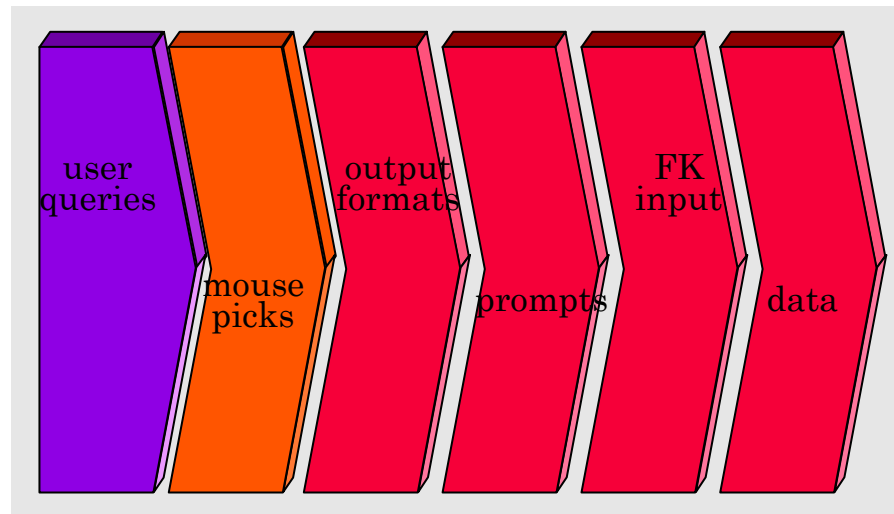
Running and evaluating the test scenarios

Measuring testing progress

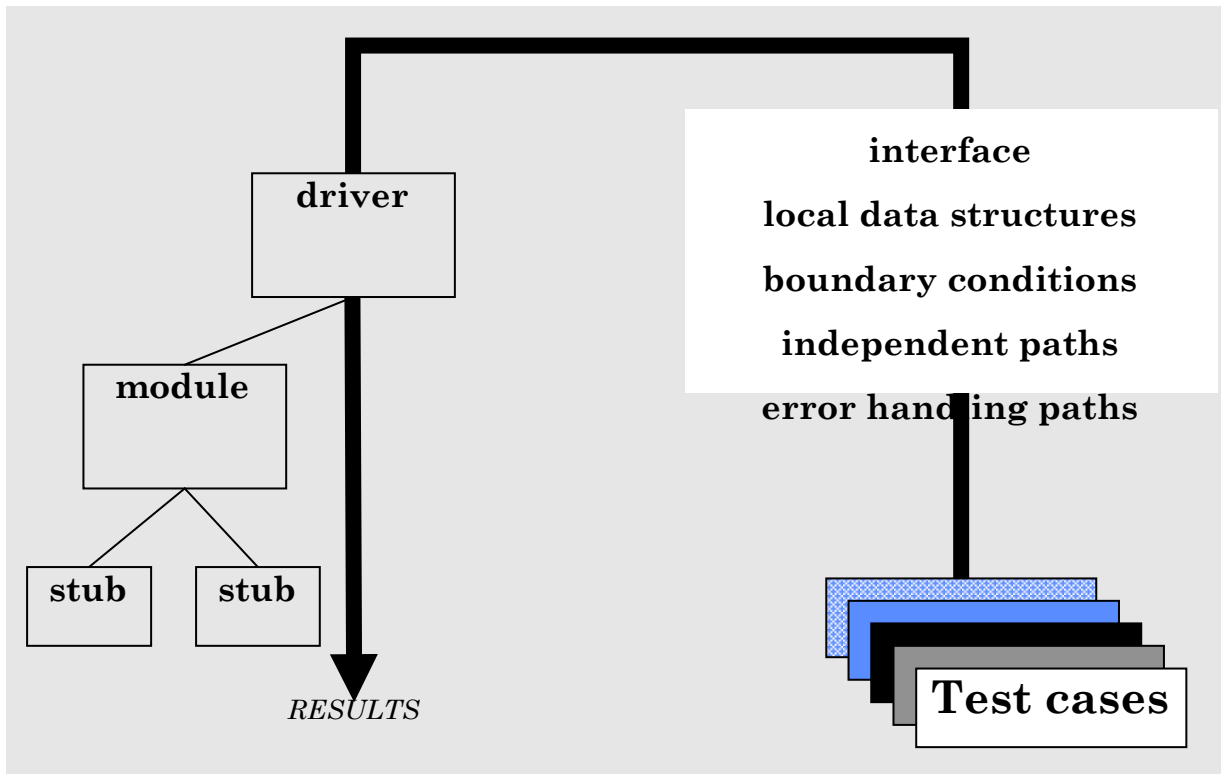
What Testing Shows



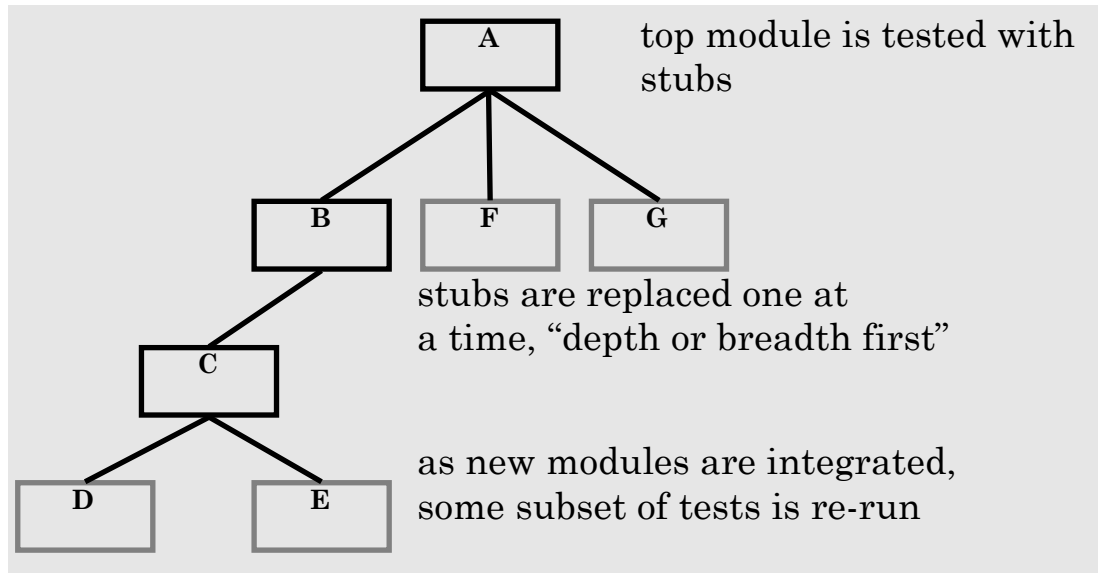
Equivalence Partitioning



Unit Testing Environment



Top Down Integration



Debugging: Symptoms & Causes

symptom and cause may be
geographically separated

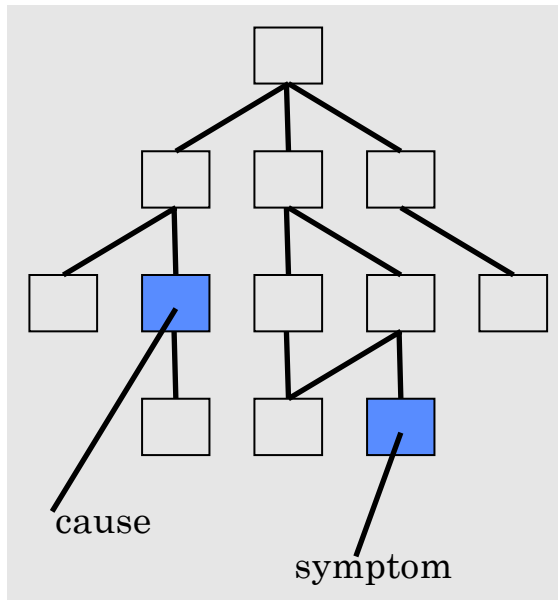
symptom may disappear when
another problem is fixed

cause may be due to a
combination of non-
errors

cause may be due to a system or
compiler error

cause may be due to assump-
tions that everyone
believes

symptom may be intermittent



Who Tests the Software?



Understands the system
but, will test "gently"
and, is driven by "delivery"



Must learn about the system,
but, will attempt to break it
and, is driven by quality

Agile Testing Case

Enterprise Information System XP Conformant

- Short Releases, System tested every two weeks, planning game, sitting together, customer collaboration, stand up meetings, continuous integration

XP Divergent

- Semi-formal specification for each feature (pair specification, ownership, and standards), QA team for acceptance testing (not customer tests), automated acceptance tests rather than unit tests

Acceptance Testing with EasyAccept

Story Test-Driven Development
Test Driven Development by Example
Client Verifiable Artifacts

Benefits

Precise and effective communication between client and developer

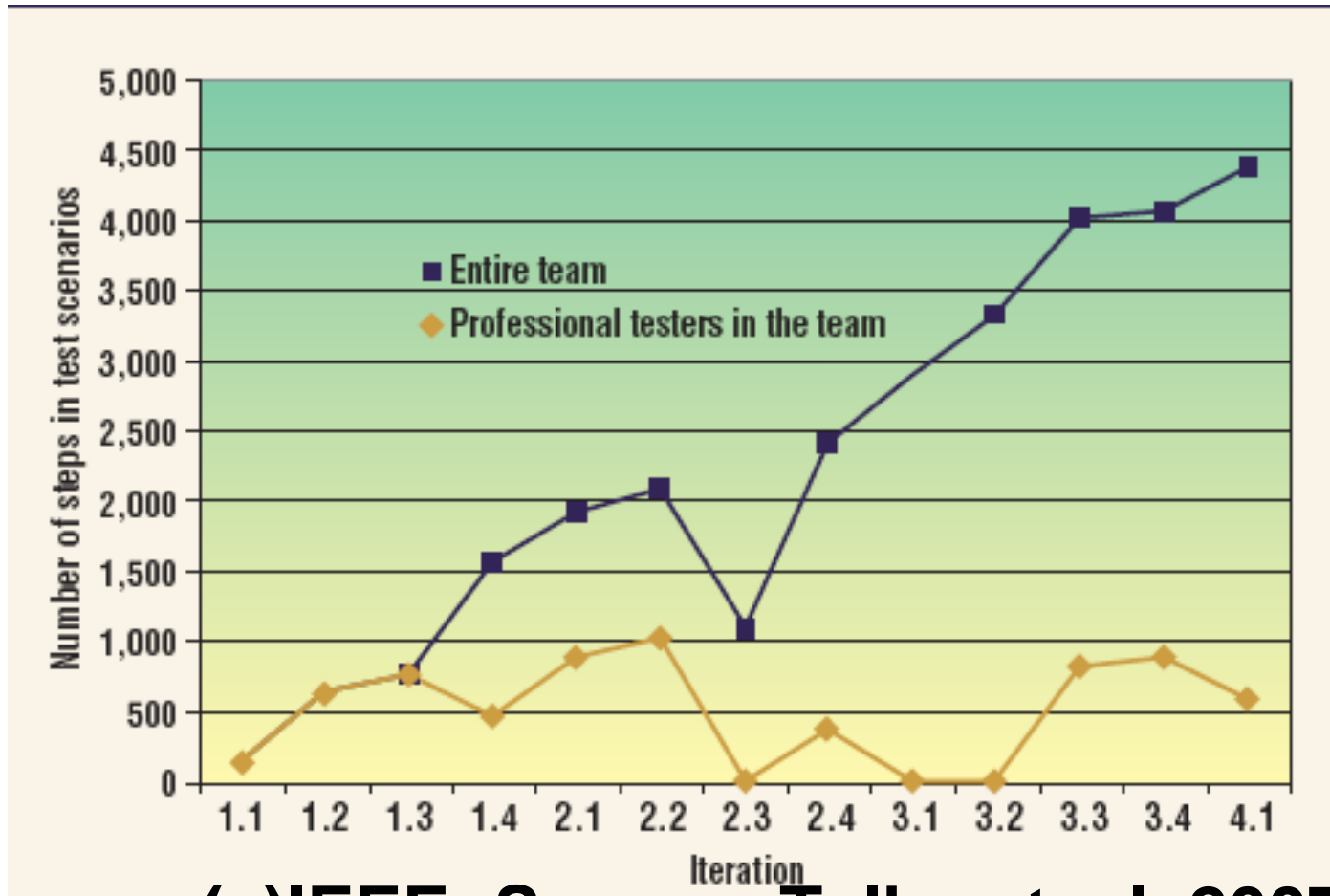
Executable artefacts for tests

Readable by clients

Quality agreements

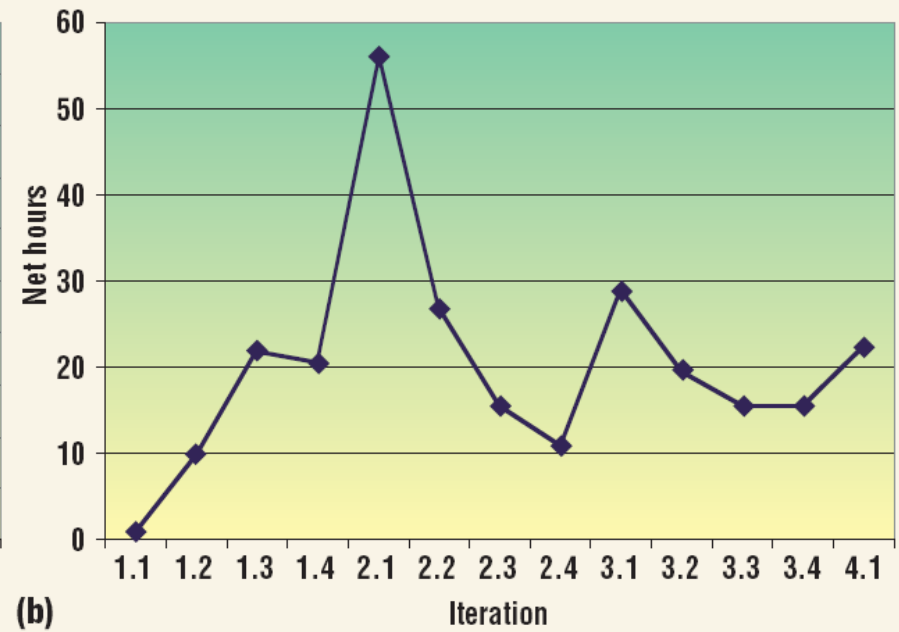
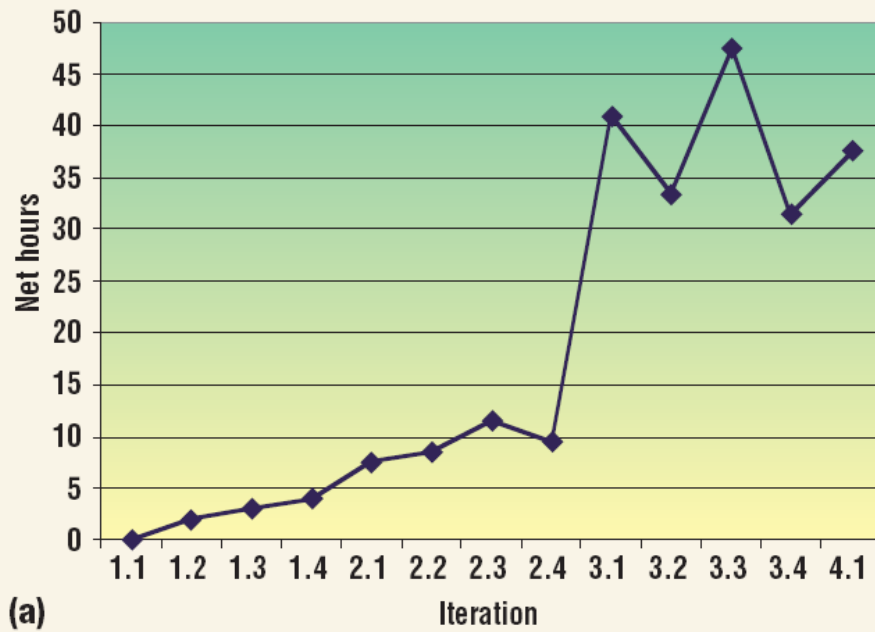
All parties know the state of the art of the features

Product Size per Iteration



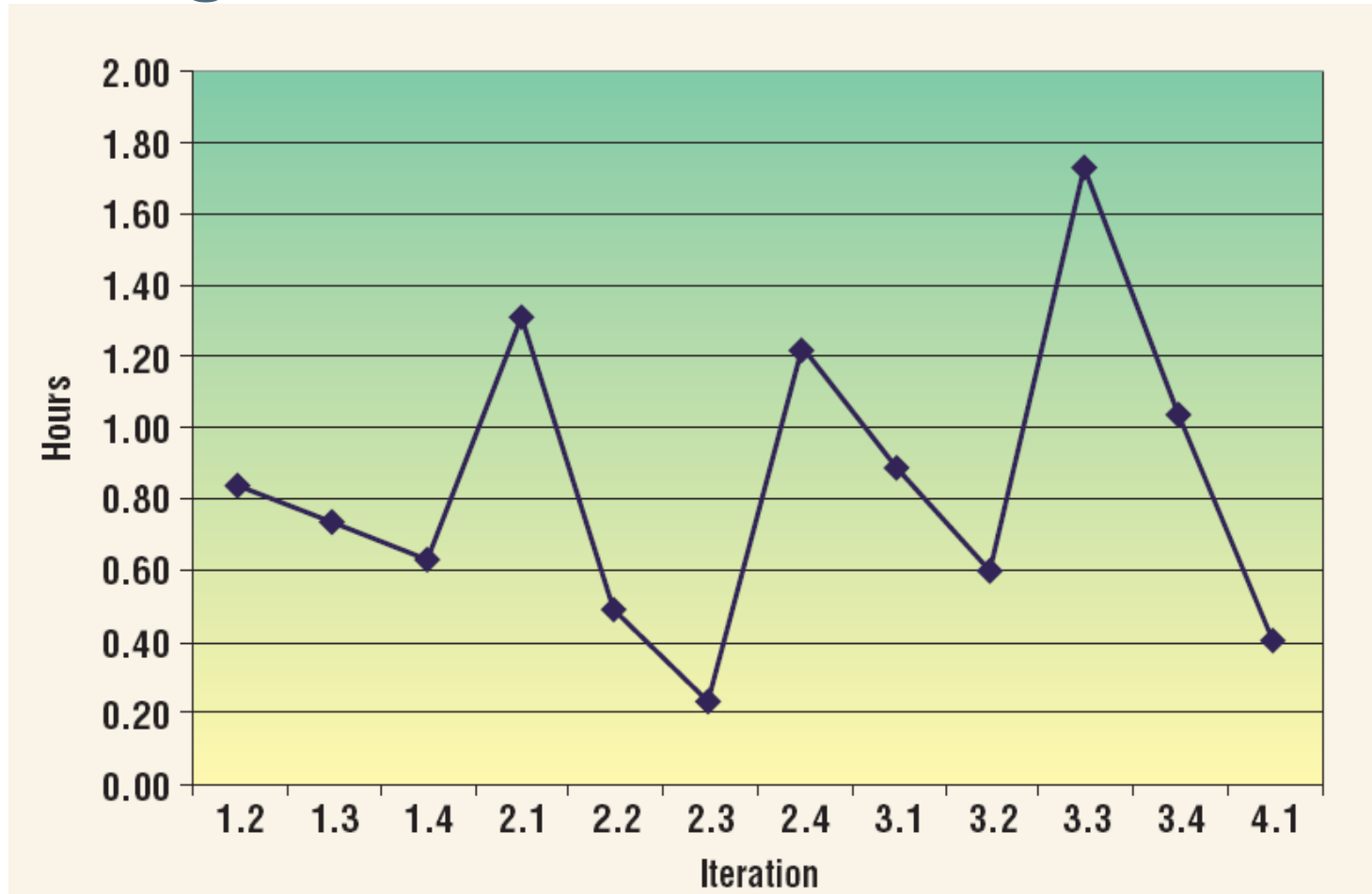
(c)IEEE; Source: Talby et. al. 2007

Testing and Defect Repair



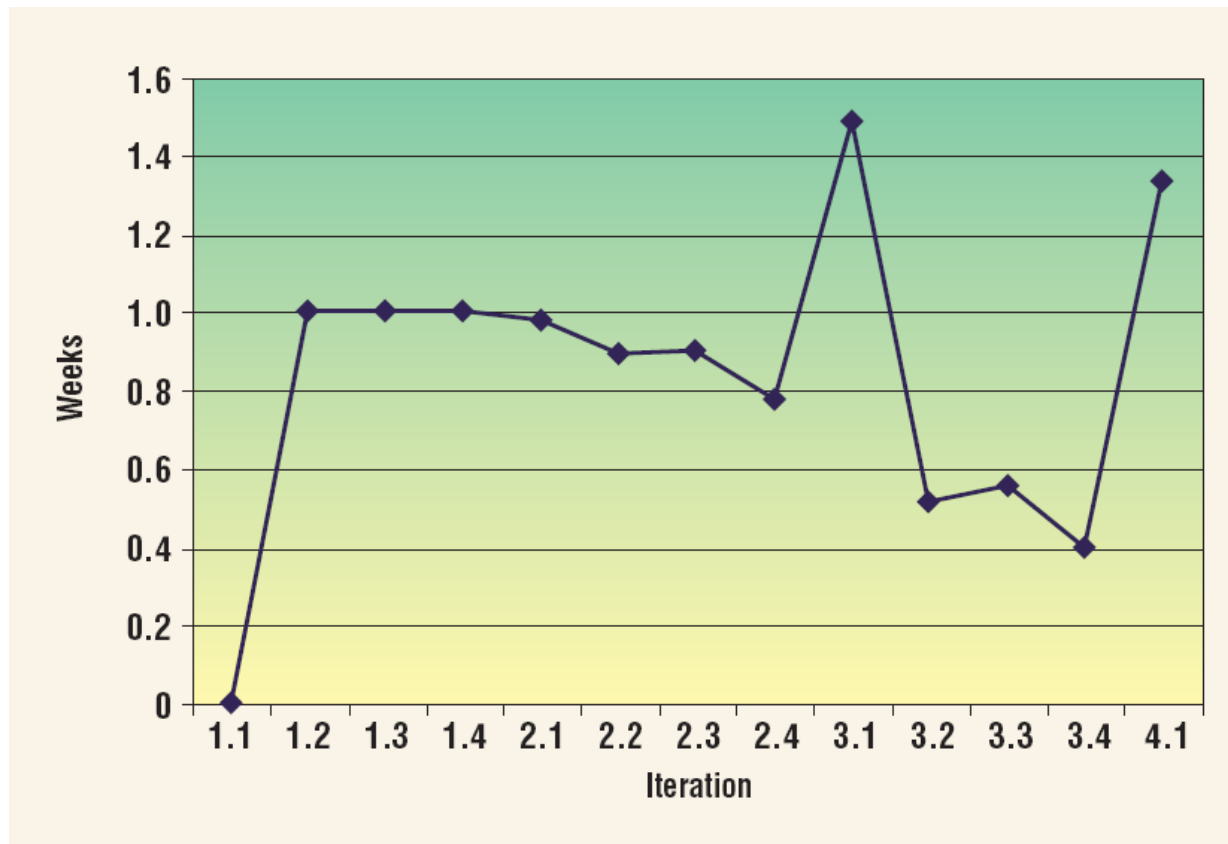
(c)IEEE; Source: Talby et. al. 2007

Average Net Time to Fix a Defect

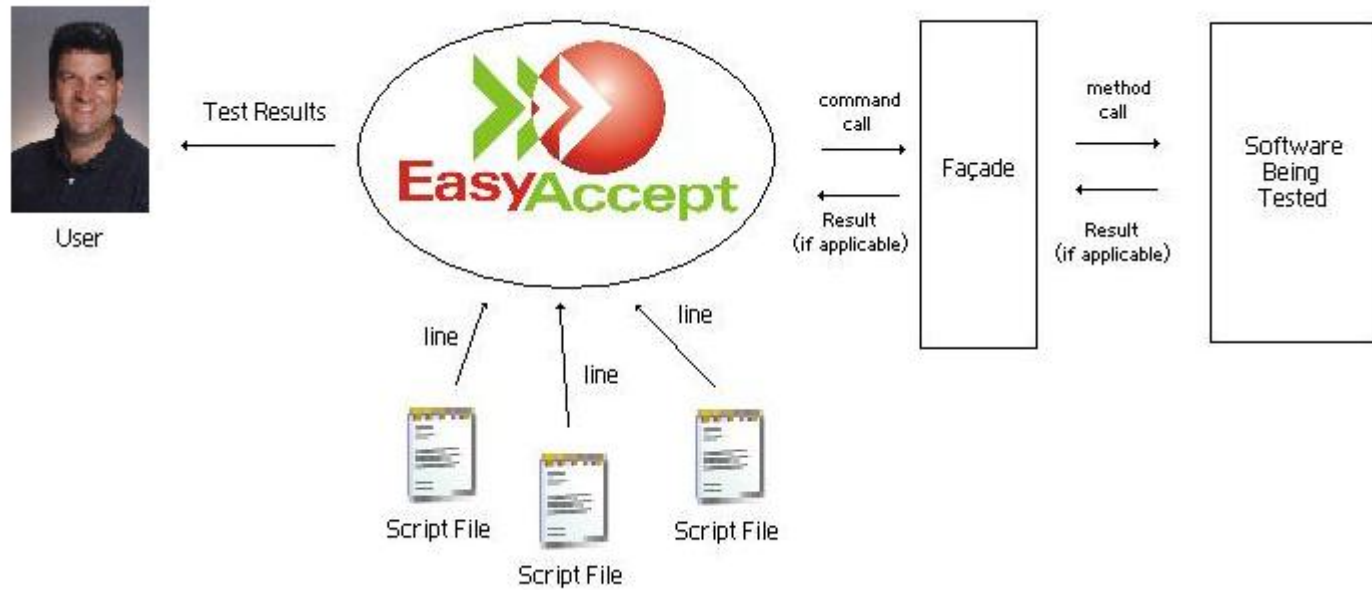


(c)IEEE; Source: Talby et. al. 2007

A Defect Average Longevity



(c)IEEE; Source: Talby et. al. 2007



<http://easyaccept.org/>

Commands

- *expect* – used to express an expected result of a command.

Example:

```
expect 5/10/1972 getBirthDate name=John
```

- *expectError* – used in situations where a command should result in an error. Example:

```
expect "There is no such customer"  
getBirthDate name=Mary
```

- *equalFiles* – used to check if two files are equal; this is useful for massive textual or non-textual testing.

Example:

```
equalFiles result.txt template.txt
```

- *expectTable* – used to do tabular input/output testing.

Example:

```
expectTable jewelName getJewelColor  
ruby red  
emerald green  
sapphire blue
```

Process

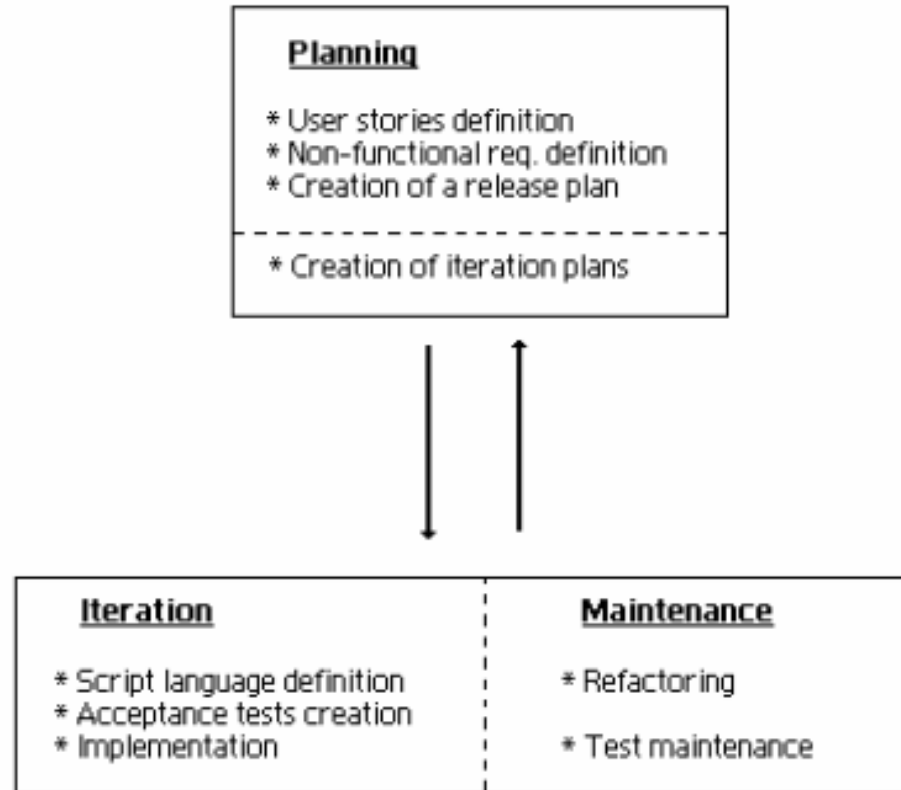


Fig. 4 – An outline of the ATDD core activities

User story: Create a New Monopoly Game

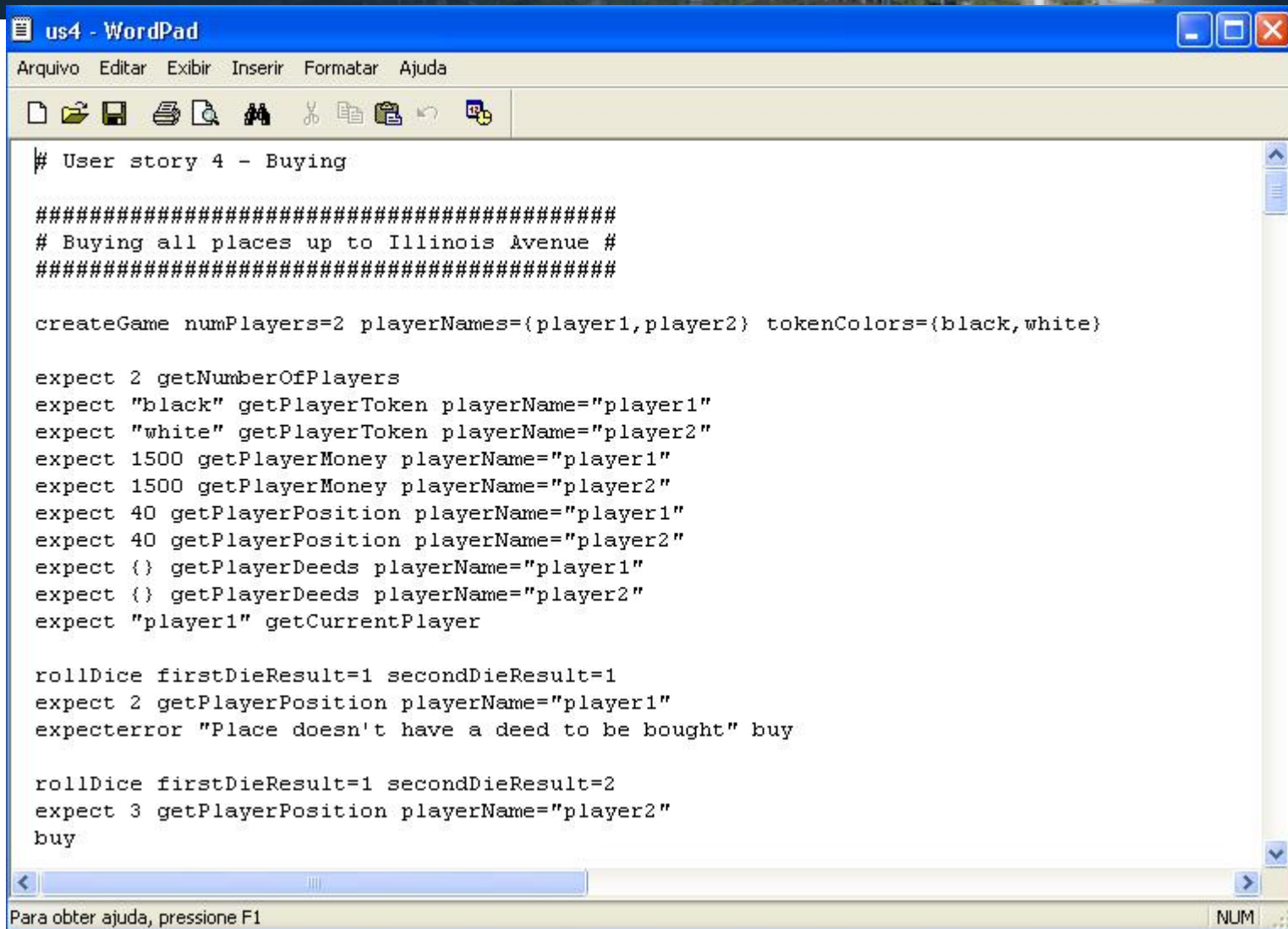
“Allow a new Monopoly game to be created. In order to create a game, users must provide the number of players, which must be between 2 and 8, a name and a token color for each player. Token colors must be chosen among the following: black, white, red, green, blue, yellow, orange, or pink. All players are placed on the first board position, labeled “Go”, and start the game with \$1500 each, and no title deeds.”

Doer: `createNewGame`

Getters: `getNumberOfPlayers`, `getPlayerName`,
`getTokenColor`, `getBoardPosition`,
`getPlayerMoney`, `getPlayerTitleDeeds`

Candidate preparers: `setPlayerPosition`,
`setPlayerMoney`

Box 1 – Translating a user story into script commands



```
# User story 4 - Buying

#####
# Buying all places up to Illinois Avenue #
#####

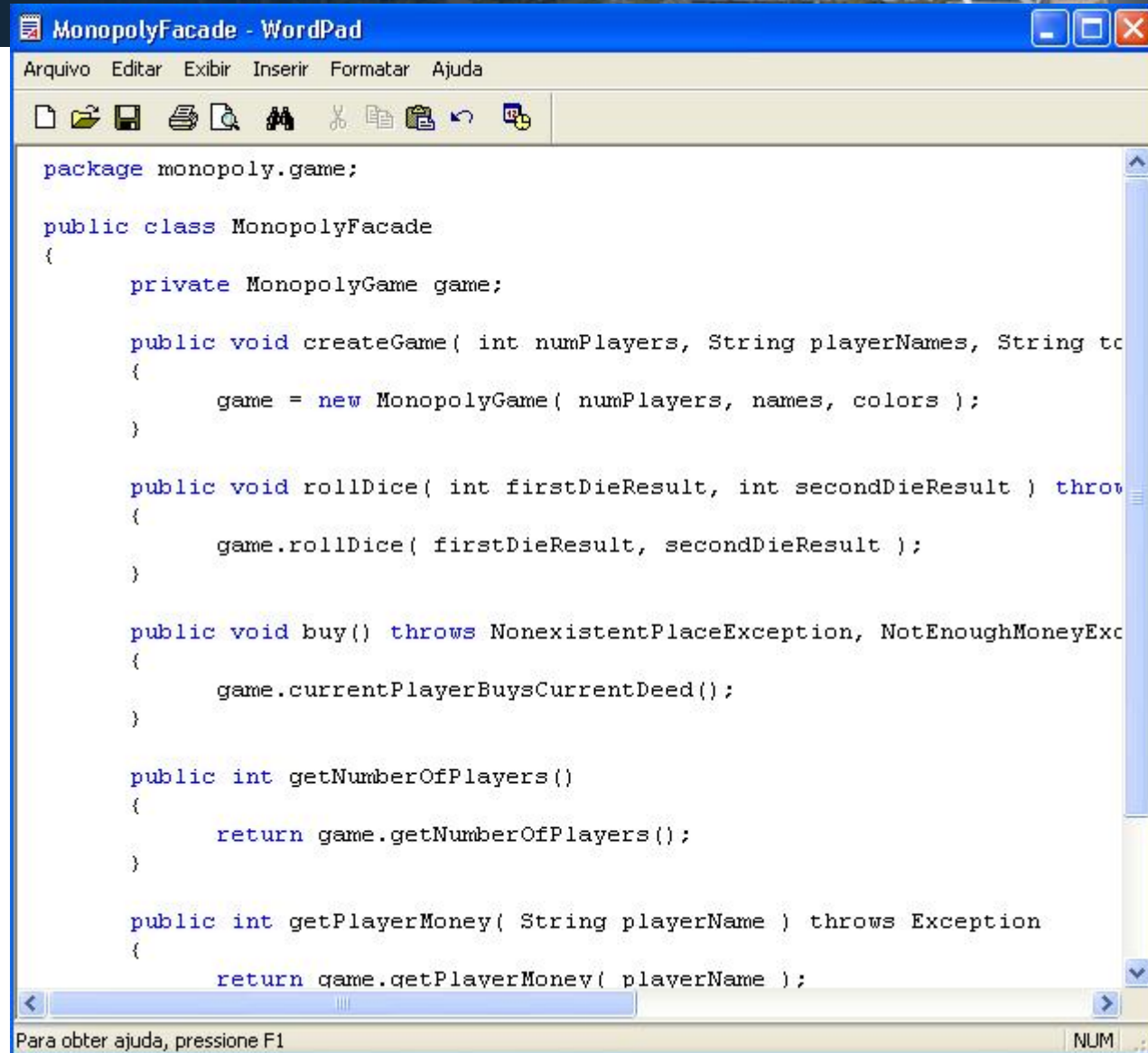
createGame numPlayers=2 playerNames={player1,player2} tokenColors={black,white}

expect 2 getNumberOfPlayers
expect "black" getPlayerToken playerName="player1"
expect "white" getPlayerToken playerName="player2"
expect 1500 getPlayerMoney playerName="player1"
expect 1500 getPlayerMoney playerName="player2"
expect 40 getPlayerPosition playerName="player1"
expect 40 getPlayerPosition playerName="player2"
expect {} getPlayerDeeds playerName="player1"
expect {} getPlayerDeeds playerName="player2"
expect "player1" getCurrentPlayer

rollDice firstDieResult=1 secondDieResult=1
expect 2 getPlayerPosition playerName="player1"
expecterror "Place doesn't have a deed to be bought" buy

rollDice firstDieResult=1 secondDieResult=2
expect 3 getPlayerPosition playerName="player2"
buy
```

<http://easyaccept.org/>



```
package monopoly.game;

public class MonopolyFacade
{
    private MonopolyGame game;

    public void createGame( int numPlayers, String playerNames, String to
    {
        game = new MonopolyGame( numPlayers, names, colors );
    }

    public void rollDice( int firstDieResult, int secondDieResult ) throw
    {
        game.rollDice( firstDieResult, secondDieResult );
    }

    public void buy() throws NonexistentPlaceException, NotEnoughMoneyExc
    {
        game.currentPlayerBuysCurrentDeed();
    }

    public int getNumberOfPlayers()
    {
        return game.getNumberOfPlayers();
    }

    public int getPlayerMoney( String playerName ) throws Exception
    {
        return game.getPlayerMoney( playerName );
    }
}
```

<http://easyaccept.org/>

Goal

Agile Testing Principles

- Testing overview
- Unit testing environment
- Large agile testing case
- Case results
- Easy accept

Tutorial on Design and XP reflections (d401a, s601d)

EVO