

# CS409

# Software Testing

**TAN, Shin Hwei**

陈馨慧

Southern University of Science and Technology

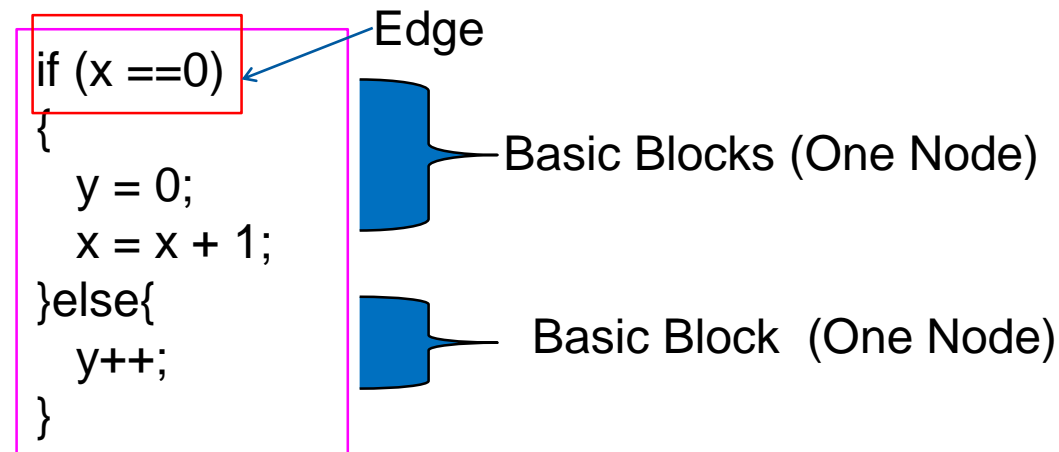
Slides adapted from cs498dm (UIUC) and cs4218 (NUS)

# Administrative Info

- No plagiarism is allowed for any assignment!

# Recap: Draw CFG Graph

- Draw one node for each basic block
- Connects basic block with edge
- Label each edge with branch predicate
- Understand how the control transfer from one stmt to the other



# CFG

- Step 1: Draw the CFG graph for onClick() in <https://github.com/BradTeachesCode/BasicCalculator/blob/master/BasicCalculator/app/src/main/java/com/bradteachescode/basiccalculator/MainActivity.java>

```
public void onClick(View view) {  
    switch(view.getId()){  
        case R.id.key_0_btn:  
            number = number + "0";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_1_btn:  
            number = number + "1";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_2_btn:  
            number = number + "2";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_3_btn:  
            number = number + "3";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_4_btn:  
            number = number + "4";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_5_btn:  
            number = number + "5";  
            answerTV.setText(number);  
            break;
```

```
        case R.id.key_6_btn:  
            number = number + "6";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_7_btn:  
            number = number + "7";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_8_btn:  
            number = number + "8";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_9_btn:  
            number = number + "9";  
            answerTV.setText(number);  
            break;  
  
        case R.id.key_add_btn:  
            symbol = "+";  
            num1 = Integer.parseInt(number);  
            number = "";  
            break;
```

```
        case R.id.key_sub_btn:  
            symbol = "-";  
            num1 = Integer.parseInt(number);  
            number = "";  
            break;  
  
        case R.id.key_div_btn:  
            symbol = "/";  
            num1 = Integer.parseInt(number);  
            number = "";  
            break;  
  
        case R.id.key_mult_btn:  
            symbol = "*";  
            num1 = Integer.parseInt(number);  
            number = "";  
            break;  
  
        case R.id.key_clear_btn:  
            symbol = "";  
            num1 = 0;  
            num2 = 0;  
            number = "";  
            answerTV.setText("");  
            break;
```

```
case R.id.key_equals_btn:
    num2 = Integer.parseInt(number);

    switch(symbol){
        case "+":
            answerTV.setText("answer: " + (num1 + num2));
            break;
        case "-":
            answerTV.setText("answer: " + (num1 - num2));
            break;
        case "/":
            answerTV.setText("answer: " + (num1 / num2));
            break;
        case "*":
            answerTV.setText("answer: " + (num1 * num2));
            break;
    }

    num1 = 0;
    num2 = 0;
    number = "";

    break;
}
}
```

# Coverage

- Step 2: Compute node coverage for this CFG
  - TR
  - Test paths

# Coverage

- Step 3: Compute edge coverage for this CFG
  - TR
  - Test paths

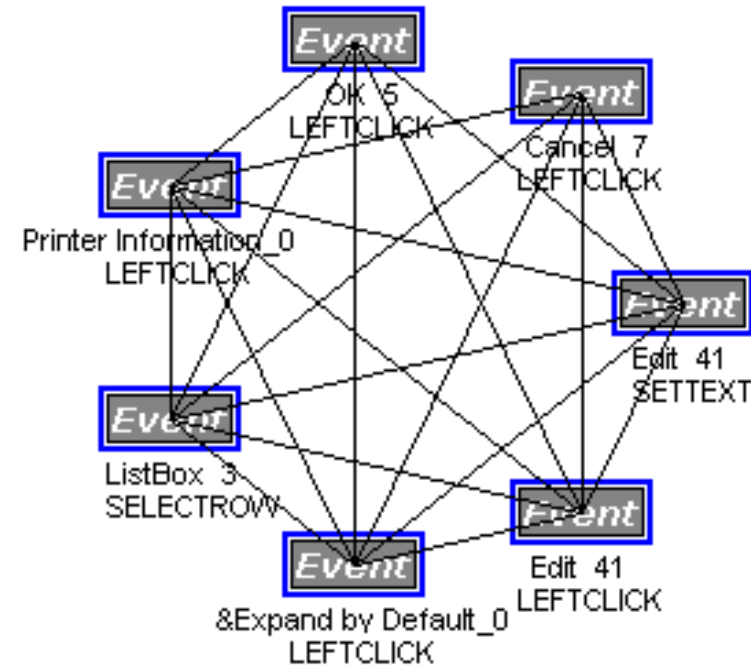


# Coverage

- Step 4: Compute Edge-Pair coverage for this CFG
  - TR
  - Test paths

# What is an Event-Flow Graph ?

- A GUI component may be represented using an Event-Flow Graph (EFG).
- EFG represents all possible interactions amongst the events in a GUI component.
- A link from one event to another means that the second event can be executed after the first event. For example, after clicking on the FILE MENU EVENT, the OPEN MENU EVENT can be executed.
- In a typical GUI component, there is a high connectivity between GUI events.



Event-Flow Graph computed for WordPad's Connect To Printer GUI Component.

# EFG

- Step 5: Draw the EFG graph for the Calculator App in <https://github.com/BradTeachesCode/BasicCalculator/blob/master/BasicCalculator/app>

# Android Graph Lab

---

# Android Graph Lab

- [https://classroom.github.com/a/-wVDOh\\_I](https://classroom.github.com/a/-wVDOh_I)

# Mobile Development

- Software development kits (SDKs) are available for most mobile platforms (Android, BlackBerry, Apple, etc.) and provide a range of developer tools necessary to create mobile applications
- Android SDK
  - Tools to build, test, and debug Android apps
  - Open source/no cost
  - [SDK download available here](http://searchsoa.techtarget.com/definition/Mobile-application-development)

# Mobile testing

- Crucial element of software development
- Application should function on devices with different:
  - Operating Systems
  - Device manufacturers
  - Memory
  - Screen resolution
  - Screen size
  - Sensor hardware
  - Types (smartphone or tablet)
  - Data Connection

# Automation tools

- Automation tools save time and money by using test scripts to repeat a test procedure multiple times
- Increase the speed of regression tests
- Examples:
  - Robotium
    - Supports Android OS
    - Automatic black box test cases
    - Free to [download](#)
  - Testdroid Cloud
    - Supports Android OS
    - Fully Automated testing on a cloud of devices
  - Calabash
    - Supports Android and iOS
    - Open source/free to [download](#)
  - Android SDK
    - MonkeyRunner



# Prepare Device/ Emulator

- Device (Phone):
  - Enable USB debugging on your device.
    - USB debugging is usually found under Developer Options, so if you have not enabled Developer Options yet, go to Settings>About Phone>tap on “Build Number” 7 times, and you will get an alert that Developer Options are enabled. You can now go into the Developer Options to turn on USB Debugging.
- Create Emulator
  - Create a new AVD using AVDManager
    - <https://blog.csdn.net/Xushuai0616/article/details/55802725>

# What is ADB?

ADB (Android Debug Bridge) is a command line tool that allows interaction with a connected device.

# Install & Run ADB from SDK


- View logs while replaying test scripts
  - Command line:
    - adb logcat
  - Android Studio:
    - Click View > Tool Windows > Logcat

## References:

<https://appuals.com/install-adb-windows-7-8-10/>

<https://developer.android.com/studio/debug/am-logcat>

To display the log messages for an app:

1. Build and run your app on a device.
2. Click **View > Tool Windows > Logcat** (or click **Logcat**  in the tool window bar).

The Logcat window shows the log messages for the selected app, as selected from the dropdown lists at the top of the window, as shown in figure 1.

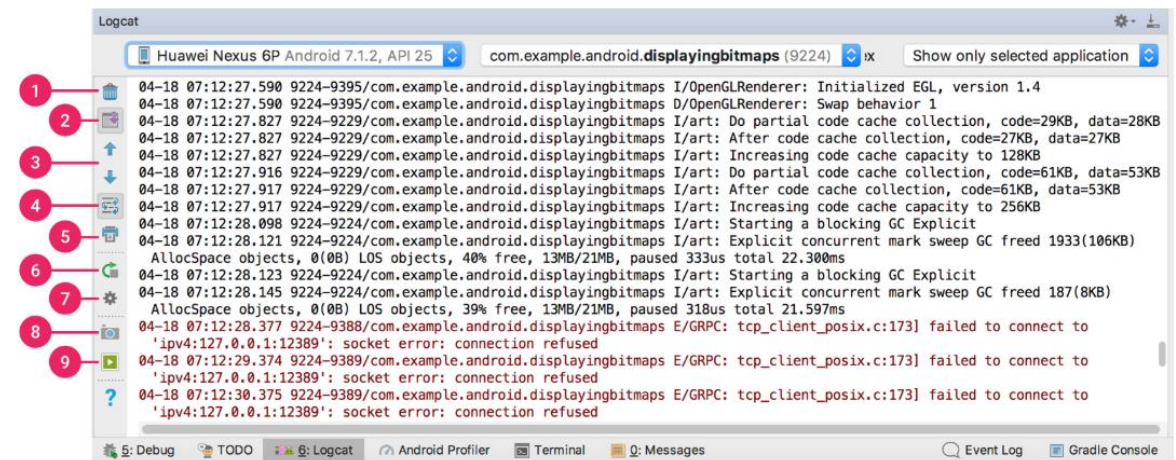


Figure 1. Logcat window

# List of useful ADB commands

- *adb install C:\package.apk* – Install an .apk package from your C:\ to your Android device.
- *adb uninstall package.name* – Uninstall an app package from your device – package name would be the specific app package name as seen in your device, for example, com.facebook.katana
- *adb push C:\file /sdcard/file* – Copies a file from your C:\ to your devices SD card.
- *adb pull /sdcard/file C:\file* – The reverse of ADB push.
- *adb logcat* – View the log from your Android device.
- *adb shell* – This will open an interactive Linux command line on your device.
- *adb shell command* – This will run a command on your device's command line.

# What is MonkeyRunner?

MonkeyRunner is a tool that provides an Application Programming Interface (API) for writing programs that control an Android device.<sup>2</sup>

- Interactive through command prompt using Jython
- Use commands to write a script

<sup>2</sup>: [http://developer.android.com/tools/help/monkeyrunner\\_concepts.html](http://developer.android.com/tools/help/monkeyrunner_concepts.html)

# MonkeyRunner Modules

- **MonkeyRunner**
  - Class of utility methods
  - Provides methods that
- **MonkeyDevice**
  - Represents a device or emulator
  - Provides methods that simulate interactions
- **MonkeyImage**
  - Represents a screen capture image
  - Provides methods related to screen captures

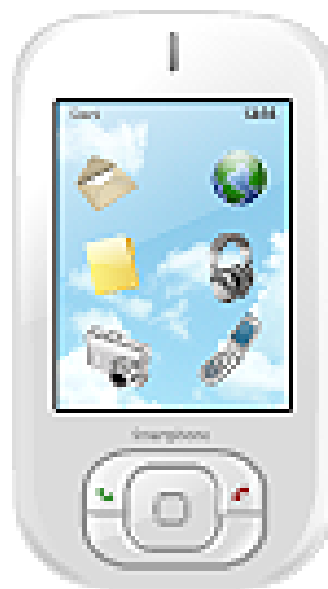
# Technical components

- Eclipse IDE
  - PyDev Extension for Eclipse
  - Jython
- Android SDK
- Android Debug Bridge
  - Command line tool that allows interaction with a connected device
- Android Package (APK) for the application to be tested
  - File format used to install applications on Android OS
- Android device connected through USB (or emulator)
- Android drivers for specific device installed on computer

# Record Test Script

## 录制测试脚本

- Usage:
  - `monkeyrunner monkey_recorder.py`



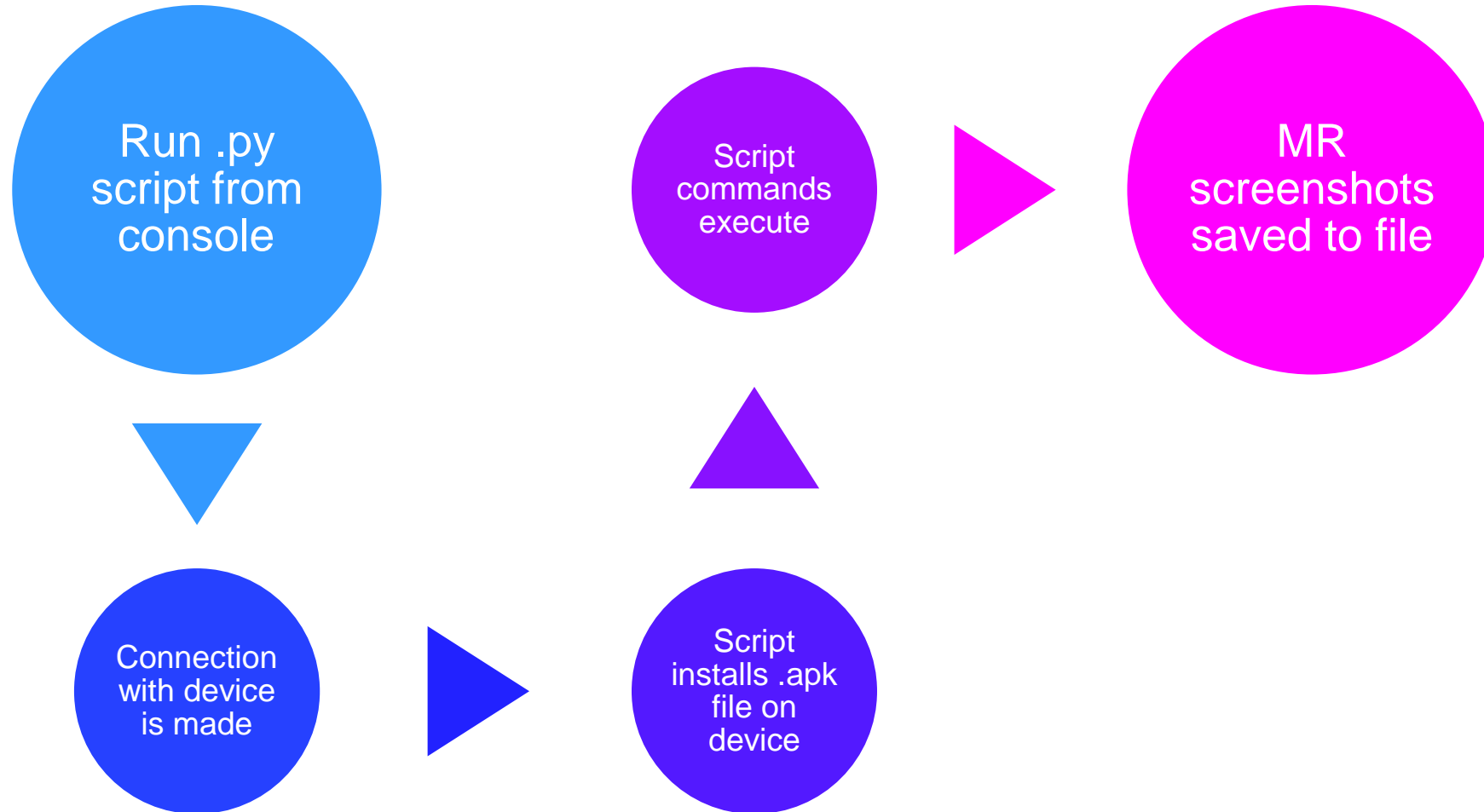


# Replay Test Script

## 重放测试脚本

- Usage:
  - monkeyrunner monkey\_playback.py <script name>





# Record and Replay

- Step 6: Record and Replay the test paths for EFG:
  - Compute
    - Node Coverage
    - Edge Coverage
    - Edge-Pair Coverage

## Repeat Step 1-5 for the selected app for class project

- Select one important method (e.g., onClick method)

# Repeat Step 6 for the selected app for class project

- Draw the EFG for your project
  - Compute:
    - Node Coverage
    - Edge Coverage
    - Edge-Pair Coverage

# What to submit?

**All answers should be submitted in README.md**

- Student name and student id
- The answers for Step 1-6 for the calculator app
- The answers for Step 1-6 for the selected app

# References

- Monkeyrunner
  - <https://blog.csdn.net/xifeijian/article/details/8580377>