Debugging Domain-Specific Modeling to Android

Hakan Ozler
ozler.hakan@gmail.com

Abstract

In this paper, it is demonstrated debugging domain-specific modeling to android devices. It is related with other programming technic. It is provided more reliable and safely way to illustrate debugging. I will demonstrate how to debug a model between ATOM^3 and Android Device. Also, it is important practice with those technic.

1 Introduction

Software development includes a huge area. Also, software development provides the latest approach which is domain-specific modeling. Domain-Specific Modeling provides to change code. It can be powerful to create source code directly from the domain-specific languages models. Domain specific modeling involves reliability of code generation. Debugging relates with domain-specific modeling with this project model. It is illustrated how to debug and why debugging is so important. Debugging process has different technics such as execution modes, steps, runtime variable, breakpoints, jump to and stack traces. In this project it will be introduces about breakpoint process by programs. Debugging checks the number of bugs or defects. It is essential to control for all code block.

Section 2 will describe related work and explain debugging with breakpoints. Section 3 will describe meta-model for the abstract syntax of RPG game and Breakpoint. Section 4 will illustrate android architecture, life cycle and some important component for project. Section 5 will explain aspect oriented programming and clarify how to correlate with android application

briefly. Section 6 will explain socket programming in python. Section 7 will clarify socket programming in java. Finally, Section 8 will illustrate an example about debugging between android device and ATOM^3.

2 Related Work

The most related article is Debugging in Domain-Specific Modeling which is illustrated useful debugging concept such as breakpoint by Hans Vangheluwe and Raphael Mannadiar[6]. Debugging is a special term of all programming languages. It is provided to fix errors, and bugs. In generally, debugging has more interactive behavior for programming languages. Moreover, Domain-specific modeling permits a special role for each debugging facility. Also, debugging is provided more performance for domain-specific modeling. It is essential to make a transform from graphical entities to the code side. There are useful semantics that is mapped onto formalism such as Petri nets, State charts, Differential equation or code. Some software industry foundation is preferred domain-specific modeling with some extra component such as debugging tool. Also, Debugging includes more efficient perform. Debugging seeks only with synthesized artifact and modeling tool. Domain-specific model is providing to set breakpoint into the domain-specific code, to pause/resume execution. In this project, it was created a breakpoint model that is considered with RPG game model. The domain-specific coding is more powerful way to provide the application logic. Debugging affords some facility on code. There are some debugging facilities features in Object Oriented languages. Those are related with popular IDEs. Language primitives are provided without any debugger process. Language primitives includes print statements, assertions, and exceptions module. Debugger primitives includes execution modes, steps, runtime variable I/O, breakpoints, jump to, and stack traces. This debugger primitives form an attractive effort for writing debugable code. Debugging in DSM includes two main subjects which are developing models and developing model transformations. Breakpoint is demonstrating to translate to the model transform debugging. Breakpoint is useful for domain specific coding to provide if there is a different or trigger on code. Respectively, breakpoint is triggered by programmers decision. It is more common sample when someone shows his own model. It is shown some breakpoint that is related with 2 meta-model objects in this project. Also it is possible to do multiple breakpoints within model.

3 The Meta-Models

In this section, it is demonstrated 3 meta-models which are RPG game meta-model and Breakpoint meta-model. Those are related each other to make a debugging on android devices. Moreover, they have a responsibility to illustrate well instruction. Those models that generate aspect file and java android file to set up infrastructure between android device and Atom3. Also, they illustrate their Entity Relationship Diagram (ERP) and Class Diagrams. Most of all, It is essential to provide domain specific modeling.

3.1 RPG Game Meta-Model

The domain-specific formalism RPG game allows users to model a game with some special constructs. RPG game model includes essential construct which are tiles, hero, villain, key, weapon, goal, trap, door, and obstacle class. These classes provide a set of the abstract syntax of RPG game model. Each construct has different attribute and behavior. It can be generated a java code after modeling a RPG game. The essence of game relates with heros heal and heros goal. If hero dies or gets goal, game is over for both of them. The meta-model of RPG game is shown as a model in the Entity Relationship formalism in Figure 1.

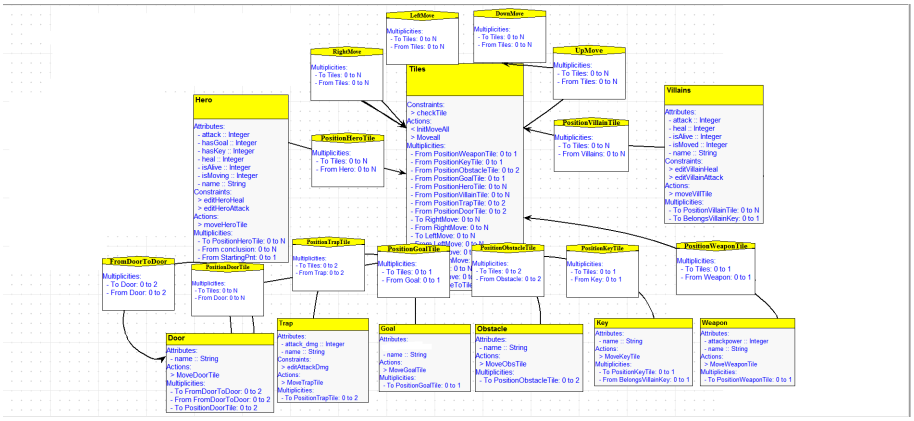


Figure 1: Metamodel for the abstract syntax of RPG Game

Tiles model can be connected to other tiles. Also, it has 4 maximum capacities since there are 4 associations between tiles which are named Left-Move, RightMove, UpMove, and DownMove. Moreover, each tile has only 4

edges. Furthermore, tiles are related to other ER model.

Hero model must be connected to tiles. It can move other tiles over tile. Also, it has 7 attributes to determine itself. Those attributes are fundamental for game progressing. Those attribute names are attack, hasGoal, hasKey, heal, isAlive, isMoving, and name. It is also combined with tiles.

Villains model must be connected to tiles. It can move other tiles over tile. Furthermore, it has 5 attributes which are attack, heal, isAlive, isMoved, and name. If it is created 2 scenes, one villains isMoved attribute should be false because there will be a clash if two villains have same value. It is also attached with association between tile and villains model.

Door can be connected to tiles. It has name attribute. It can be attached at most one door. If RPG game includes two scenes, door has to be attached to one door in order to movement of hero.

Trap can be connected to tiles. There are special attributes for traps. For instance, attack damage which is triggered when hero goes to traps tile, and also name.

Goal can be connected to one tile. It is special for games rules. It has trivial attribute which is name. Moreover, it has an association with tile. *Obstacle* can be connected to other tiles. It includes more advantage from others For instance, when hero wants to go obstacles tile, hero cannot move there. It has trivial attribute which is name. Moreover, it has an association with tile.

Key can be connected to tile. It has a high priority for hero. Truly, if hero moves doors tile and he has not a key to open it, he cannot cross other scene without key. There is also trivial attribute which is name. Moreover, it has an association with tile.

Weapon can be connected to tile. It has a private attribute which is attackpower. For instance, when hero moves weapons tile and gets the weapon, his attack will increase according to weapons attackpower attribute.

In addition, there is a button, which includes on RPG game meta-model,

to generate code from meta-model graph to android platform. It is represented that RPG game objects are transformed to java objects. This button that creates a java file, which name is MainActivity.java, generates a main activity class for android application and also includes rpg game class. It is more convenient way to transform the RPG game formalism.

3.2 Breakpoint Meta-Model

Breakpoint formalism is provided for debugging between ATOM^3 and Android Application. It has two ER models that are named Server and Breakpoint classes. Moreover, it possesses two buttons which includes in button models. First button provides to generate aspect file for RPG game, which is named aspectRPG.aj and is saved in main direction of atom3 folder. Second button provides to start a server into the RPG game model and it listens on port after users give a port number via server (Server model will be explained).

It is explained two ER- models that involve important entities: Breakpoint

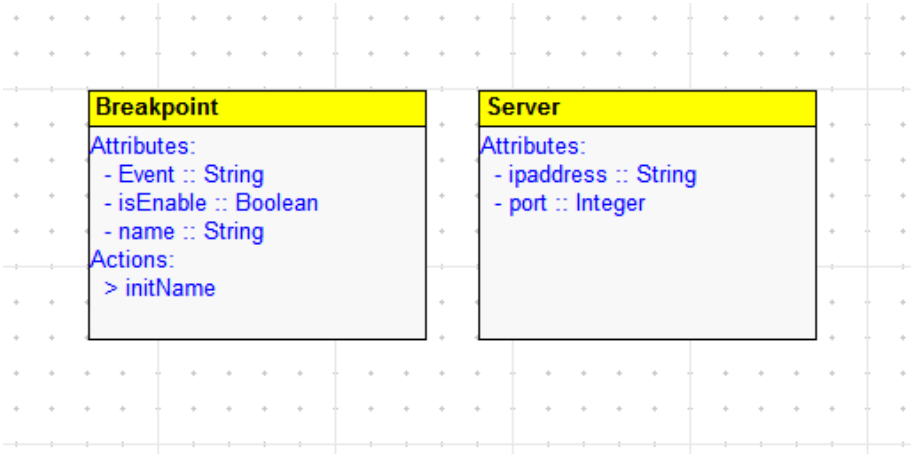


Figure 2: Metamodel for the abstract syntax of Breakpoint

can be connected to RPG game entities to make a debugging during execution. It is important think to correlate with RPG game and breakpoint structure. It has three attributes that provide breakpoint behavior of the game.

- *Event* attribute is important, when RPG game catches a breakpoint , users can see which breakpoint was triggered and show breakpoints event name.
- *isEnable* attribute is secured, whether breakpoint is available or not.
- *Name* attribute is a trivial component like other entities.

Server cannot be connected to other entities. It must be included on RPG game model to connect with an android device via internet. Moreover, it establishes an infrastructure of the socket component. When users add server entity for RPG game, there will be a server on users game. It has two major attributes to open a socket for waiting incoming message.

- *Ipaddress* attributes must be empty. It means that the empty string is used for the interface address, allowing incoming connections from any interface on the host.
- *Port* attributes should be 4 any number. A port is identified for each address. In this project, it is given a port number that is 8888. Then, users can press second button which is Start Server button, to listen incoming message on port number via Socket.

Start Server button provides a communication between ATOM^3 to Android application. It waits an information message which is related with details of game, from android device when a game catches a breakpoint during execution.

4 Android

Android was coded by Google and Open Handset Alliance. It is open source, multiprocessing, and multithreaded OS. Google launches the code under the apache license. They preferred Linux-based operation system because it is more secure, provides more library options and memory usage and processing well formed

4.1 Android Life Cycle

Each application runs in its own process. Each activity of an app is run in the apps process. Processes are started and stopped as needed to run app components. Usually, certain management of the life cycle is done automatically via activity. The activity class has the essential methods callbacks to help users manage the app. Those control flows are sequantial for each activity class:

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onRestart()
- onDestroy()

4.2 Android Applications

Android applications are defined to Android in Eclipse (IDE) [5]. There are two main components for each application that are named AndroidManifest.xml, layout.

- Android Manifest file which must include for each applications in its root directory.The manifest which provides user permissions, activity, service, provider elements, presents usefull and important information, about the application to the Android system.
- Layout is tipically a xml file. It can be named different name for applications. Actually, it presents the visual structure for a user interface (UI). Users can change some screen elements or elements attributes in xml file. It would be easy way to demonstrate all component in the same place.

5 Aspect Oriented Programming

AspectJ was released at 1998 at Xerox PARC. It is an extension of Java. AspectJ is an implementation of aspect-oriented programming (AOP) for Java. It is a new way to improve the separation of concerns of programs. AOP languages typically extend object-oriented languages. The AspectJ is distinct and open source, it is very mature. AspectJ runs with Eclipse and other IDEs. It is an essential to indicate some decomposition crosscutting concerns much like object-oriented programming. There are several new constructs which are pointcuts, advice, inter-type declarations and aspects. The main constructs are pointcuts and advice that can modify a program's control flow and state. Aspect construct encapsulate all of these references. Also an aspect is like a Java class, but adds support for pointcuts, advice and inter-type declarations. Pointcuts describe a set of join points; for instance: a method call, method execution, retrieving fields, etc. AOP provides to separate each concern [3]

There are two main problems that aspect oriented programming tries to solve, which are code tangling and code scattering. Code tangling means that one module includes many concerns. Code scattering means that one concern is scattered across several modules in a project. in project. AOP tries to collect the concern in one aspect interface. Section 4 will explain that how aspect can works with android application

5.1 Aspect Oriented Android Development

Android is commonly in Java, it does not run Java bytecode. Android uses the Dalvik VM. Accordingly, it uses a completely different bytecode than java. However, the Android SDK includes the dx tool to translate java bytecode to dalvik bytecode, which is why users are able to write Android applications in java. It is generally thought of as a java virtual machine. On the other hand, aspect oriented programming compiles classes and aspects to java bytecode. Then an AspectJ Weaver weaves the Aspect bytecode into the Class bytecode. The class files are changed to Dalvik bytecode using the Dex tool.[4]
If programmers want to use aspect oriented programming in their project, they should pass some steps to put it on android project.

1. Programmer has to download The AspectJ Development Tools plugin to Eclipse (IDE) [7].

2. Programmer should be aware of some important things to develop aspect oriented [4]

3. Finally, programmer has to do some essential tasks to run this aspect tool [3].

6 Socket Programming in Python

Python is very popular programming language. It also has some advantages over the scripting languages. It has a simple syntax and is conceptually clear, making it easy to learn. Socket programming is not a difficult to understand architecture. Programmer can create your server and client easily. Python offers two basic sockets constructs that are Socket, provides standard low-level networking interface, and SocketServer, provides classes that simplify the development of network servers. Socket module includes several class methods and instance methods[8]. In this project, it is implemented server socket with some instance methods which are:

- sock.bind((address,port)) (sock is an instance of socket. socket(socket.AF INET , socket.SOCK STREAM)). It represents that bind the socket to the address and port.

- sock.accept(), it represents that returns a client socket with same address imformation.

- sock.listen() , it represents that places the socket into the listening state.

- sock.connect((address , port)) , it represents that connects the socket to the defined port and host

- sock.recv(buffer length), it represents that receives data from the socket up to buffer length size.

- sock.close() , it represents that closes the socket.

Actually, socket is truly the same with other programming languages. It is a network system between the two applications that communicate each other. If programmer wants to create a socket, he/she would be create a instance of socket class which is:

sock = socket.socket(socket.AF INET, socket.SOCK STREAM). A socket object is returned. The AF INET symbol means that programmer requests an Internet Protocol (IP) socket, IPv4. The SOCK STREAM symbol represents the transport protocol type (TCP socket).

After programmer created a server socket, he / she uses the bind method to bind an address to it, the listen method includes the size of listening state, and the accept method represents to accept a new client via network. In this project, it was implemented a server socket to communicate with android device. RPG game is the server side of this project. Simply, a server socket is shown below:

```
sock = socket.socket ( socket.AF INET , socket.SOCK STREAM )
sock.bind( ( ' ', 2525) )
sock.listen( 5 )
newsocket, (host , port) = sock.accept()
```

7 Socket Programming in Java

Typically, it is the same function with java language. A socket is an endpoint of a two-way communication link between two programs running on the network. Java provides a set of classes, defined in a package called java.net. It exists the two key classes from the java.net package which are ServerSocket and Socket. The socket abstraction is very similar to the file concept. It is indicated a client socket to send an information about the newest details of rpg game to ATOM[3].

Firstly, programmer must open the socket object. He / She creates an instance of PrintWriter stream which is one of the character-based classes. It involves two parameters that are OutputStream and Boolean autoFlush attributes. Now, it is ready to send a message via PrintWriter to server socket.

Afterwards, socket should be closed after the process finished. Simply, a client socket is shown below:

```
Socket s = new Socket ( MainActivity.ipaddress , MainActivity.portn );
PrintWriter outp = new PrintWriter ( s.getOutputStream() , true);
outp.println (event);
s.close ();
```

8 Simple Debugging Example

This example demonstrates the meta-models entities such as rpg game and breakpoint model to be generated to java code for running on android device. Moreover, RPG game that is run on android device sends a message to breakpoint server if there is any breakpoint that is triggered. Firstly, it should be created a huge RPG game which includes one hero, two villains, three obstacles, one weapon, one key, two doors, two traps and one goal object. Furthermore, there are four breakpoints that are related with some entities:

1. The first breakpoint is for between Hero and Villain. This means that if hero or villain encounters in the same tile, they will attack each other and this breakpoint will be triggered from android device and will send a message to ATOM^3 via socket. Breakpoints event name is Attack.

2. The second one is for between Hero and Key. This means that if hero moves keys tile and gets key, this breakpoint is triggered and give a prompt about breakpoint. It will send a dump to server. Breakpoints event name is Getkey.

3. The third one is triggered when hero goes to other scene through the door. If hero has key to open first scenes door, then he can teleport to the second scene. Breakpoints event name is Teleport.

4. The last one is triggered when hero goes to goals tile and finds a goal. Also, this means that game is over. After hero get the goal, relevant breakpoint is triggered and send a dump message to rpg game model. Breakpoints event name is Getgoal.

After making an adjustment about breakpoints, the rpg game needs a server. It is created one server which includes ip address that is empty and port number that is 8888. In this game, there must be at most one server to start listening on port 8888

Until now, it was created the rpg game with breakpoints and server. Now, it is time to generate android java file. All entities from model are transforming to java object and they are copying their values to java file. Java file is saved into main atom3 folder. It represents as MainActivity.java. Lets give some details about the MainActivity.java.

In java file, there is a main activity class which is extended to Activity class. All elements that include on android interface, bind into the onCreate() method. It includes a RPGame class that contains all functions and methods about the rpg game. The RPGame class is implemented to runnable class. It acts like a thread and it is working separately from the main thread. Multi-Threading act 2 majors advantages. Threading provides a useful abstraction of concurrent execution and applications responsibilities can be separated. Moreover, it is used handler class to communicate back with the main applications thread through a handler object. Normally, background threads are not allowed to interact the main activitys view. It has to be used a handle object to communicate with base thread. There are also two main classes to establish objects which are named Cast and object. The hero class is a cast. On the other hand, hero is a type of the cast object. It is the same for object class. Lastly, main activity file includes a breakpoint class to save how many breakpoints exist

It was explained some useful details about MainActivity.java. The breakpoint model exists two buttons (those were explained in section 3). Users have to generate an aspectj code into the same direction of mainactivity.java. This aspect file is a hard-coding. It is running with main activity file. Also, it is checking to send a message via socket before/after some methods start or/and end. If one advice is triggered, it will send a message both main thread and ATOM^3 via socket.

Now, I can put these two files into my android project and then those will be building. There would be no any error and I will start to run my rpg game in my android device. Moreover, I will also press the start server button to

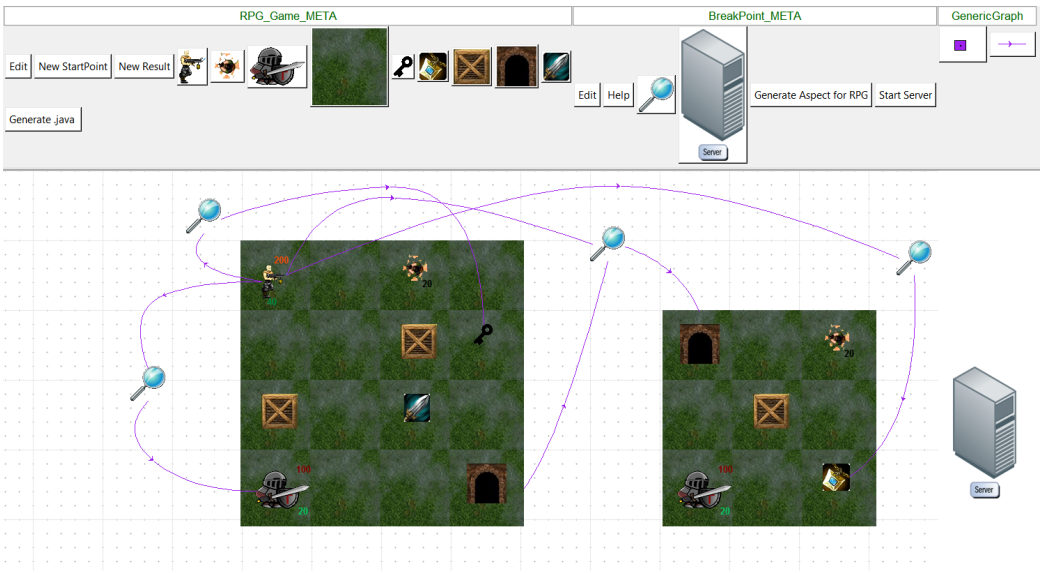


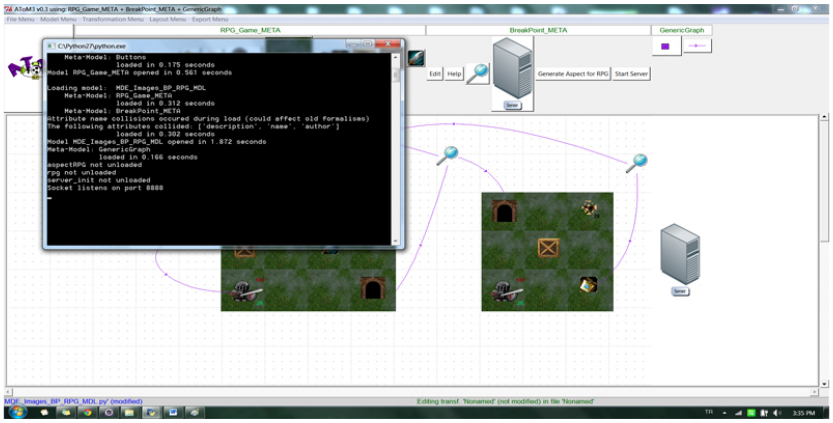
Figure 3: RPG game model and Breakpoint model

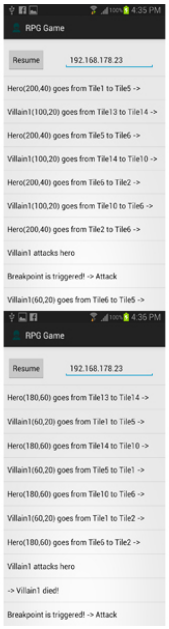
listen incoming message from device. Figure 3 shows the RPG game model and Breakpoint model in ATOM^3. My RPG game is waiting a new incoming message from devices. As I told before, I opened my android application and I wrote my PC Ipv4 address to send a message then, I pressed the run button. Figure 4 shows the android application interface. In RPG game, hero moves randomly, for this reason some breakpoints may trigger or not during the execution. Firstly, rpg game is running in sequence, it can be seen all step in android application. During this execution, game caught a breakpoint which name is Attack event. Attack event means is triggered when villain and hero attack each other. Afterwards, game stops and it sent an information details of rpg game such as:

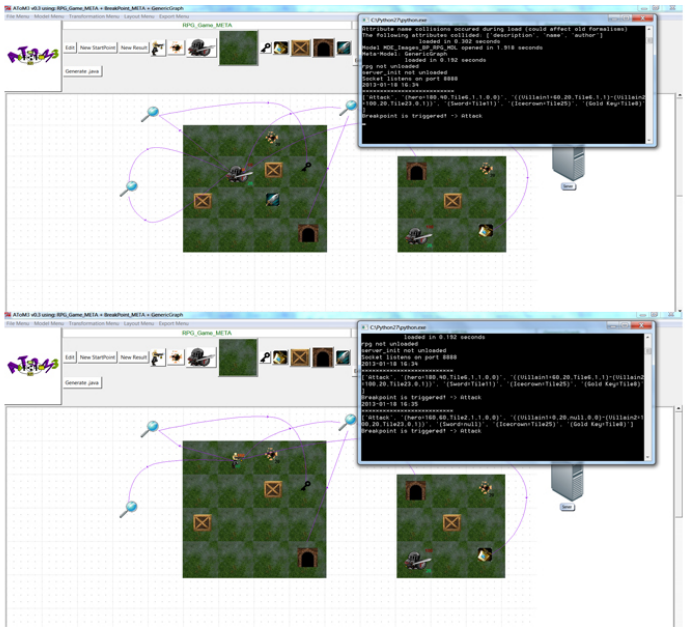
Xin=160,40,Tile4,1,1,0,0; (Villain1=60,20,Tile4,1,1 -
Villain2=100,20,Tile14,0,1); Sword=Tile3; Icecrown=Tile18;
Gold Key=Tile5.

Figure 4: Android Application Interface

Then my rpg game changed immediately. After changing, I pressed resume button and game resumed. Then again, game caught a new breakpoint. Breakpoint event name is Attack. Hero and villain attack each other and it gave a new message. Third breakpoint is Getkey. We have an other breakpoint between hero and key. When hero got key, our breakpoint was triggered and our game again stopped. Fourth breakpoint is Teleport event. Hero decided to cross other scene and our breakpoint was triggered suddenly. Until hero crossed the second scene, hero got key because otherwise hero coudn't cross the second scene. Afterwards, hero also got a weapon and his attack power increased. Futher, he skirmished with trap during the execution. All important information was sent via socket and our rpg game model changed with the new informations. Finally, last breakpoint never triggered, as i said before game runs randomly. Our hero died without reaching a goal and it was not sent a modified information to rpg game model. Figure 5 shows all situation about the game sequentially.




RPG Game
Resume
192.168.178.23
Hero(200,40) goes from Tile1 to Tile5 ->
Villain1(100,20) goes from Tile13 to Tile14 ->
Hero(200,40) goes from Tile5 to Tile6 ->
Villain1(100,20) goes from Tile14 to Tile10 ->
Hero(200,40) goes from Tile6 to Tile2 ->
Villain1(100,20) goes from Tile10 to Tile6 ->
Hero(200,40) goes from Tile2 to Tile6 ->
Villain1 attacks hero
Breakpoint is triggered! -> Attack
Villain1(60,20) goes from Tile6 to Tile5 ->
RPG Game
Resume
192.168.178.23
Hero(180,60) goes from Tile13 to Tile14 ->
Villain1(60,20) goes from Tile1 to Tile5 ->
Hero(180,60) goes from Tile14 to Tile10 ->
Villain1(60,20) goes from Tile5 to Tile1 ->
Hero(180,60) goes from Tile10 to Tile6 ->
Villain1(60,20) goes from Tile1 to Tile2 ->
Hero(180,60) goes from Tile6 to Tile2 ->
Villain1 attacks hero
-> Villain1 died!
Breakpoint is triggered! -> Attack



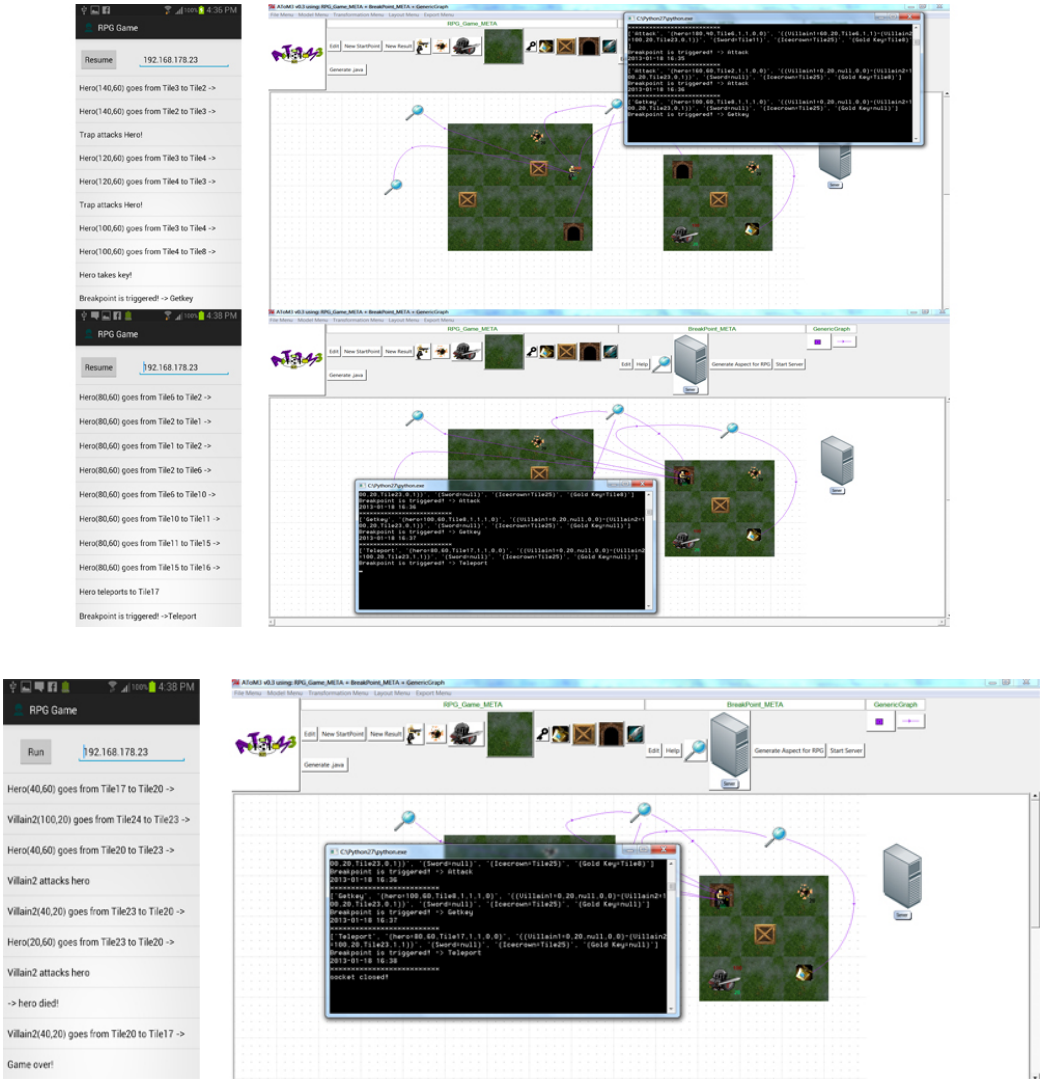


Figure 5: Demonstration of Debugging

9 Conclusion

Debugging provides useful technics. Debugging is a special term of all programming languages. It is provided to fix errors, and bugs. In addition, there are many technologies that are used to demonstrate debugging domain-specific modeling. The AToM^3 tool provides a specific modeling for us. In this project, it is demonstrated essential technic such as socket programming python and java, aspect oriented programming, and Meta-modeling. It is important to extend this project in the future work. Also, it can be used different technic with debugging. Moreover, this project can be initiated for other researches.

References

[1] Python, *Python Website*,
http://python.org

[2] AToM^3, *AToM^3 AToM3Programming WebSite*,
http://atom3.cs.mcgill.ca/people/jlara/AToM3Programming/index.dtml

[3] *Aspect Oriented Programing*
http://www.eclipse.org/aspectj/doc/next/progguide/index.html

[4] *Apect Oriented Android Development*
http://deansserver.co.uk/ dean/2011/07/18/aspect-oriented-android-development-tool-integration/

[5] *Android Guide*
http://developer.android.com/guide/components/index.html

[6] Raphael Mannadiar, Hans Vangheluwe,
Debugging in Domain-Specific Modelling

[7] *AspectJ Development Tools*
http://www.eclipse.org/ajdt/

[8] Sockets programming in Python *M. Tim Jones, Sockets programming in Python,04 Oct 2005*