By Robert Ryan McCune

September 2007

WIPER Documentation

How To Run the WIPER Simulations

The Wireless Phone Based Emergency Response (WIPER) System, originally developed by Tim Schoenharl, is an emergency response system designed to provide emergency response managers tools to respond more effectively and efficiently to crises. The simulation component of the system is integral in the detection and recognition of emergencies in near real-time. The purpose of this document is to provide step-by-step instructions for any user so one may run and further develop the WIPER Simulations.

The following instructions are the steps taken by the author to run the WIPER Simulations and are oriented toward a beginner, unfamiliar with any of the environments. As a disclaimer, I concede that several, more efficient methods surely exist to successfully compile and run the simulations. However, the intention is simply to get the simulations up and running. For example, while Linux is not entirely necessary, my work took place on the Ubuntu Feisty Fawn OS and as such I recommend anyone foreign to programming to work on a Linux distribution. Following these instructions word for word will allow individuals of varying computing backgrounds to run the simulations.

Several open source toolkits and programs are required for the simulations to operate. To begin, make sure your system runs the most updated version of Java, currently JRE 6.0. Then download Eclipse, a programming environment easily compatible with the WIPER Project, at eclipse.org, being careful to choose the correct distribution for the appropriate operating system. Proceed to download Repast J 3.1 from repast.sourceforge.net. Repast is an open source agent-based modeling toolkit ideal for the objectives of the WIPER Simulations. Currently, the most recently released Repast Simphony Alpha versions do not support the WIPER System's GIS maps and other functionalities, so the appropriate download is the older and more developed Repast 3.1 toolkit.

Also, if not already completed, download the WIPER source code from nd.edu/~rmccune , or alternatively nd.edu/~tschoenh for an earlier version. To access the source tarball, one must log into his or her Notre Dame AFS. Log into webfile.nd.edu and select the New Link button near the top, and choose the first option of a folder link. Select AFS as the folder and enter the NetID rmccune to create a link to Robert McCune's public folder. After adding the link, the user will return to the previous window, where a new folder will appear on the left of the screen (unless you have previously linked to a public AFS folder). Within the AFS Public contents will be the WIPER_Project tarball that will include this very document, source code, and required data files.

By now the user has downloaded the most recent version of Java, Eclipse, Repast, and the WIPER source files. The only items left are the GeoTools Libraries, necessary to manipulate GIS and Shapefile map data, as well as Maven, an open source project management tool that easily tracks all of GeoTools dependencies. GeoTools is available at geotools.codehaus.org, where one may download the libraries. Maven is available at maven.apache.org. Instructions better than I may provide for installing and running GeoTools with Maven through Eclipse may be found at:

http://docs.codehaus.org/display/GEOTDOC/03+First+Project

Following the entire user page will leave the user with a simple working GeoTools project in Eclipse to display the version of GeoTools. Keep the project open, for the project will be linked with our WIPER Project to easily utilize the GeoTools libraries without repeating the process.

With a GeoTools project functional, create a new Eclipse project that will serve as the WIPER Project by selecting File > New > Java Project. Enter the desired name for your project, such as WIPER_Project. Then select the Next button (not Finish) to arrive at the Java Build Settings page. First choose the Projects tab and add the aforementioned GeoTools project in order to easily include the GeoTools libraries and Maven in your buildpath. Next, choose the Libraries tab and select Add External Jars. Navigate through the proper directories to locate the Repast .jar files. Within the Repast J folder, there will be a single repast.jar file in the main directory as well as several .jar files inside the lib folder. Make sure to add them all. Finally, choose the Order and Export tab, and check every box except JRE System Library. Click Finish to create the WIPER Project.

The WIPER Project is within the Eclipse environment, albeit empty. The appropriate libraries have been added to the project and now the WIPER files will be imported to the project. Select File > New > Package and name the new package 'wiper'. Once created, right click on wiper and choose Import. Choose General > File System and navigate to the appropriate WIPER source code directory. Select the file named src, where the main WIPER java files are located (e.g. WiperSimModel.java) as well as four subfolders: moveModel, actModel, util, and test. Make sure all the .java files are included and select Finish. The package wiper will now include the main files, and four more packages should be created from the subfolders, labeled exactly as wiper.moveModel, wiper.actModel, etc. If the subfolders are not labeled properly, the user must properly rename them by right clicking and selecting Refactor > Rename. Once all the source code is imported, create two more packages named 'data' and 'log'. Follow the same process to import the data and log files from the WIPER_Project folder. If there are any errors in importing a few of the files, click OK and ignore. Once all the files have been imported to the Eclipse project, choose Project > Clean and then right click on the folder and select Refresh, or just press F5. The project should be error free, although warnings are permissible. The WIPER Simulation ready to run.

Select Run > Run from the toolbar. If Eclipse prompts the user as to which file to run, choose WiperSimModel as it is the main program. After a few seconds the Repast toolbar will appear at the top and the Runtime Window on the right. The user can input the desired runtime settings, such as Movement Model and Activity Model number, the number of simulated steps, and the proper Shapefiles to read for the map data. The Movement Model and Activity Model Types are as follows:

Movement Model 0 – Flee Model, agents flee from a point

Movement Model 1 – Random Movement Model, random agent movements

Movement Model 2 – Null Movement Model, used when testing activity model

Movement Model 3 – Flee Visibility Model, Agents appear to move but do not

Movement Model 4 - Bounded Flee Model, flee from a point to certain distance

Movement Model 5 – Move and Return, agents go to and from work

Movement Model 7 – Flee to Road, agents flee from a point, and upon reaching a road follow the road system away from the point

Activity Model 0 - Always call, agents always call

Activity Model 1 –Distribution-based calling, empirically based calling activity

Activity Model 2 – Null Activity, used when testing movments

For further information I refer the user to the source code and suggest experimenting with different classes and functions to gain more familiarity with the different elements of the simulation.

Although errors may be scattered throughout the running of the simulations, if they do not interfere with the Repast visuals, i.e. the simulation can complete a run, then the errors or warnings may be ignored. Once the simulations have been successfully run, the user is ready to take the next step and further develop the code. I highly recommend combing the provided Repast website as well as the source code for a better understanding of the different Repast functions. Happy coding.

Should any questions arise feel free to contact me at rmccune@nd.edu