**Table of Contents**

Summary (Introduction) 1

Goals and Objectives 5

Project Timeline 9

Project Development 13

Appendix 1: Usage Manual 20

Appendix 2: Acknowledgements 21

# Summary (Introduction)

My capstone project was to develop and deploy a framework for integration of a Real Time Digital Simulator (RTDS) and the DETER experimental framework for the Trusted Cyber Infrastructure for the Power Grid (TCIPG) research lab. TCIPG conducts research and experimentation into trusted systems and security for emerging smart grid applications. The integration framework provides a means by which the capabilities of the power grid simulator will be available to experiments modeled within DETER.

**Project description**

The software developed for this project is meant to be a framework that connected two of the TCIPG research lab’s more important resources. The Real Time Digital Simulator (RTDS) provides researchers with the capabilities of simulating very large power grid infrastructure. These simulations enable researcher to perform experiments that would otherwise be impossible. The DETER experimental framework provides a common set of tools and utilities for performing reproducible and redistributable experiments. By combining these two resources, research will gain the ability to perform research and experimentation on large power grid simulations in a reproducible way.

The integration framework developed for this project provides the mechanism to connect these two resources together. By providing an easy to use API that experiment developers can use to implement data gathering and simulation control, as well as a means to interact with that data within DETER, the researcher will be able to design and execute experiments that take advantage of both resources.

Development of the project was done in three phases. Each phase was meant to address specific needs of the researchers and development team at different point along the development time line. This three-phase approach was instrumental in giving research developers on the KERK project the necessary tools to demonstrate their work for industry partners and other research associates. Additional phases provided reasonable milestones to make development assessment about the progress of the project as well as to provide well-defined “break points” for the reevaluation of project requirements.

The first phase of the project consisted of developing a prototype that was specifically tailored to the demonstration needs of the KERK project research developers. As noted in the proposal, the time to completion for this phase was very short due to an upcoming scheduled demonstration. While the shot timeline required some late working nights, the prototype was completed on time, and the demonstration was presented to academic and industry partners with great success.

The second phase of the project focused on code cleanup and applying sound programming practices. As with any prototype, the programming required to prove the application concept, the code for this prototype was a patchwork of large functions, and inconsistent application of programming best practices. In this second phase, the key logic of the framework was isolated, and the surrounding support code was refactored to apply accepted python and object oriented best practices.

Upon complete of the second phase, the integration framework was still specific to the KERK requirements. The third phase addressed the need of the larger lab and other researchers who will be using the lab. In this phase, the requirements gathered from researchers beyond the KERK project were addressed. The second phase release of the integration framework served as a model of the type of experiment that will be performed using the integrated systems as well as the primary test case.

Once the development of the integration framework was completed, work for finalize the integration of the new framework with the deter framework was started. Unfortunately, due to shifting priorities, only the first step of this multi-step process was completed. Following discussions with research partners and considering the established priorities of the TCIPG research lab, other projects were shifted to higher priorities. Completion of the integration work will recommence upon completion of the other high priority projects near the beginning of 2013.

**Review of other work**

The primary sources of related works were the documentation and manuals for the Real Time Digital Simulator and the accompanying software RSCAD as well as the documentation and a code review for the DETER experimental framework.

In addition to the simulation design and execution features of RSCAD, it also implements a simple scripting interface. This scripting interface provides two key features that used to implement the integration software. Specifically, RSCAD provides a ListenOnPort function that causes the scripting environment to open a network socket and listen for incoming network connections. This connection can be used in much the same way as the RSCAD provided editor, in that an RSCAD script will be read from the network connection and executed as if it were composed in the RSCAD script editor.

The other important RSCAD feature is the ListenOnPortHandshake function. This function provides a simple synchronization interface that can be used to ensure that both the script environment that is executing the script and the remote client component are assured to be at the same synchronization point before moving on to execute subsequent tasks.

**Rational and system analysis**

As the TCIPG research lab primarily investigates power grid security concerns and power grid related technologies, the ability to simulate real-world power grid applications is one of the lab’s highest priorities. In addition, like all experiments, the results must be reproducible, by other researchers to validate the findings of the experiment. The two, related, technologies in this project provide those two key attributes to the lab.

Operating on a real power infrastructure is infeasible due to cost and the expertise required to operate it safely. The RTDS provides a means by which complete power infrastructure can be simulated, in real-time, and accurately substitute for real infrastructure. And, while, the results from a simulation are easily reproducible, the RTDS is unable to simulate the experimental, and often completely new, concepts and ideas that are the focus of much of TCIPG’s research.

DETER provides a means by which researchers can implement these new ides in both cyber and physical forms, and conduct experimentation on these ideas. DETER gives researchers the resources necessary to create reproducible experiments, and collect results than can be passed on to other researchers for verification.

In addition to the project goals, this project also fulfills one of the general goals for the TCIPG research lab. The Lab’s mission is to provide the tools, equipment, and personnel necessary to conduct both cyber and physical power grid security research. By integrating these two devises we move one step closer to developing a world-renowned tool kit to be used for power grid security research.

# Goals and Objectives

For this project, there were three project specific goals, and one organizational goal. The project goals were to give researcher programmers the ability to control the power grid simulator from with their research programs, to enable the replay of previously gathered data, and to integrate the new integration framework with the DETER framework. Finally, this project meets one of the ongoing lab goals. This project provides one, of many, professional quality tools to be used by researchers to research power grid security.

**Controlling the Grid Simulator**

The primary goal of the project was to give researchers programmatic control of the power grid simulator. During normal operation of the simulator, a researcher was required to run their simulation, collect data from that simulation, and then perform experiments and investigative research on that data. This defeats the point of the real time simulator. This project connects the simulator to the DETER experimental framework, and give researchers the ability to perform experimentation on a real time data flow.

Following initial documentation review, it was discovered that the simulator’s companion software, RSCAD, provided two function that that could be used to facilitate the necessary functionality. RSCAD provided a scripting environment for performing programmatic interaction action with the simulator. By connection this scripting environment to DETER, it becomes possible to achieve the first goal.

Further research, as well as direct communication with representative from RTDS Technologies, the manufacturer of the simulator. A plan was developed to use the ListenOnPort script function to make the RSCAD script environment remotely accessible. ListenOnPort causes RSCAD to listen for incoming connections from third party software. Over this connection, script commands can be sent to RSCAD.

This functionality was used to connection the new integration framework to the RSCAD script environment. The integration framework includes an ability to load and send RSCAD script to RSCAD over this network connection, giving the researcher the ability to control the simulation and collect data in real time. Through the use of the integration framework, this can be done from within the DETER framework.

Finally, RSCAD provides a ListenOnPortHandshake function. This function is used to provide synchronization points between the experiment and the simulator. For instance, when initially loading a simulation, the load operation can be performed, and ListenOnPortHandshake called. The experiment can then wait for notification from RSCAD that ListenOnPortHandshake has returned before continuing. This ensures that the simulator is in a specific, predictable state prior to continuing to subsequent stages of an experiment.

**Replay Collected Data**

As previously mentioned, researchers normally use previously collected data when performing experiments. During the requirements gathering, it became clear that researchers wished to be able to continue to use this method of accessing data. This is especially important during the development of an experiment. While experimental results should come from a real time data stream, the simulator is a “scarce” resource. That is, there is only one simulator. By comparison, the lab is capable of supporting many experiments, and researchers desired the ability to develop and perform initial tests even when the simulator is not available.

In addition, as the simulator is a scarce resource, the design of the integration framework allows for multiple, simultaneous, output targets. By enabling the use of multiple targets, the researcher can perform tests using live data, while collecting the data stream to be used later during development. Further, if two experiments require similar data, then the multiple target feature can be used to collect a data stream for other researchers, while performing live experiments.

The multiple target feature was developed by allowing the data stream to be sent to an experiment, as well as write the data stream to disk. The integration framework can also read a data stream from disk, and send it to a target (another disk stream or an experiment, or both). This feature gives researchers the ability to use the generated data stream for testing new experiments as well as during the development process for experiments.

The final direct project goal was to integrate the new RTDS integration framework with the DETER experimental framework. In order to be a successful project, the integration framework must be usable within DETER. The original plan called for a full integration of the integration framework into the DETER user interface as well as add configuration and control components as an extension to the NS2 language used by DETER. By doing so, when a researcher designs their experiment, access to RTDS becomes a natural part of the experiment design.

Unfortunately, full integration was not completed. Due to higher priority tasks, lab leadership concluded that a partial integration was acceptable, and full integration can be added as a later task.

The partial integration took the form if adding the integration framework to a specific DETERS image type. By doing so, the integration framework become available to any experiment that includes that image type in the experiment definition. To configure and run the integration framework, a researcher will be required to start the framework, either directly or though the start-up command interface in DETER, and the simulator separately, and “hook up” the plumbing required to connect their experiment to the framework.

This method of connecting an experiment to the simulator requires a little more work on the part of the research programmer, however, by following the partial integration route, lab personnel are freed up to work on higher priority projects.

After discussing the altered plan with TCIPG research programmers, it was discovered that such a move was an acceptable compromise, as the higher priority lab work was also highly desired by the research programmers. Note, this change was not discussed with the researchers as the change has very little impact on them.

**Professional Tools**

Finally, one of the goals of the TCIPG research lab is to provide professional high quality tools to developers and researchers for conducting power grid and power grid security related research. The DETER experimental framework is one part of that goal, as is the grid simulator. By connecting these two technologies, we give researchers one more toll to conduct reproducible experiments that explore the power grid and security related to the power grid.

Many efforts are underway to develop new tools and add resources to the lab to create a best-in-class research facility that is available to the power community, as well as to serve as the model for other such facilities. This project serves as a demonstration of interfacing two dissimilar technologies in a fashion that eases development of research projects and applications.

While this is only one tool, now available for general use in the lab, further development will continue in an effort the build the definitive research facility for researching power grid related technologies.

# Project Timeline

Task Name Duration Start Date End Date

Planning

* **KERK Interviews** 1 Day 10.31.2011 11.01.2011

A meeting was scheduled with the KERK research programmers. As the KERK research programmers and I TCIPG development team, these interviews were conducted during our regular weekly team meeting. Familiarity with the KERK project on my part eased communication of requirements. The requirements were approved on the spot, and development commenced the same day.

* **General Interviews** 3 Days 11.14.2011 11.17.2011

Over the course of three days, TCIPG researchers were contacted and interviewed. The result of these interviews was that the KERK requirements went beyond the general needs of all of the researchers interviews. As a result, the final decision was to use the KERK requirements as both the KERK specific version of the requirements as well as the general requirements.

* **Develop Execution plan**
  + **KERK** 1 Day 11.01.2011 11.02.2011

With the assistance of the KERK research programmers, an execution plan was developed. It was decided that, due to a very short timeline, the KERK demonstration and prototype would be developed concurrently. The KERK research programmers would use the prototype as it was being developed, and report bugs, or request features as necessary.

* + **General** 2 Days 11.18.2011 11.20.2011

The general execution plan entailed refactoring the prototype to transform t into a professional quality tool.

**Phase One Development**

* **Develop KERK requirements document** 1 Days 11.01.2011 11.01.2011

The KERK requirements document was developed in coordination with the KERK research programmers. This was done during a regularly scheduled development team meeting. Developing the requirements document in this fashion trimmed a couple of days for the expected timeline. This was beneficial as it added buffer time to the development stage, which was already subject to a short, hard deadline.

* **Obtain KERK researchers approval** 1 Day 11.01.2011 11.01.2011

Again, final approval of the KERK requirements document was obtained during a regularly scheduled development team meeting. Doing so meant that development could start on the prototype immediately, ahead of schedule by two days.

* **Develop KERK targeted prototype** 8 Days 11.01.2011 11.08.2011

As has been mentioned, the prototype completion was subject to a hard deadline. Specifically, the prototype was required to be operational and work, as needed by the KERK demonstration, by 12.00 PM November 8. The KERK researchers were scheduled to demonstrate their project at 2.00 PM on that day. The demonstration was given to TCIPG industry sponsors.

To meet the tight deadline, it was necessary to spend time out side of normal working hours to complete the prototype. In coordination with the KERK research programmers, the prototype was completed the day prior to the deadline, making it possible to perform the demonstration for a camera crew. The video of the demonstration was presented to non-sponsor members of the TCIPG research community.

Task Name Duration Start Date End Date

* **Industry Day Demo** 1 Day 11.08.2011 11.08.2011

The industry Day KERK demonstration was not, technically, part of this project, it did server as a demonstration of how this project is expected to be used in the research lab. It also served has a hard deadline for the completion of the prototype. As noted, meeting this deadline required several long days, but the deadline was met, and the demonstration was successful. A video of the demonstration is available on the Internet at <http://www.youtube.com/watch?v=BCNjIzvc_eg>

**Phase Two Development**

* **Refactor KERK prototype targeting** 5 Days 12.01.2011 12.08.2011  
  **company defined coding practices**

The first stage refactoring started on schedule. The primary goal of this refactoring step was to identify the key algorithms and business logic, and isolate them into functions and classes that can be reused for the final version of the framework. As expected, there ware no surprises during this phase. The resultant code was clean and complied with company coding practices and, as it was written in python, the python PEP8 style guide.

**Phase Three Development**

* **Develop general requirements document** 5 Days 12.08.2011 12.15.2011

As noted above, following the researcher interviews, the researchers and I concluded that the KERK requirements met and exceeded the general requirements. As such, the KERK requirements become the final general requirements

* **Obtain general researcher approval** 5 Days 12.08.2011 12.15.2011
  + Note: Holiday schedule is reflected in due dates

Due to the decision to use the KERK requirements as the general requirements, researcher approval was obtained during the interview process. Again, upon closer examination of the KERK requirements, the TCIPG researchers concluded that it was not necessary to develop an additional set of requirements. These conclusions were made during the interview process, and approval of the KERK requirements as the general requirements was obtained at that time.

Task Name Duration Start Date End Date

* **Develop final application and framework** 10 Days 12.15.2011 12.25.2011

Development of the final version of the integration framework was started a full month early. This was due, primarily, to the decision to use the KERK requirements as final general requirements. Using the business logic and algorithms identified during the first refactoring stage, the framework was rewritten with an eye the generalizing the features and making the interaction API easier to use. This was completed successfully, taking the predicted amount time to complete.

**Documentation**

* **Develop API documentation** 2 Days 01.5.2012 01.10.2012  
    
  The API documentation is meant to be a reference for research programmers when developing experiments that use the integration framework. Again, there were no surprises during the development of the documentation. Special thanks goes to a fellow researcher who provided editing and proofreading of the documentation. See Appendix 3 for API documentation.
* **Develop user manual (canceled)**  
  Following discussions with researchers and research programmers, it was concluded that the built in help messages were sufficient to provide all necessary usage documentation. See Appendix 2 for usage documentation.

**Project Review**

* **Review final generated artifacts (excludes** 1 Day 01.12.2012 01.12.2012  
  **intermediate artifacts such as the KERK  
  prototype and stage one refactor)**Upon completion of the project, and as is expected practice, the project code underwent a code review. This ensure that other developers have had a chance to weight in on the used coding practices, and ask any questions that may com up. In addition, this code review ensures that all hard requirements are met, and that the code can be released for general use in the lab. This review process was completed on the day the review was requested with no additional comment from the reviewer.

Task Name Duration Start Date End Date

* **Documentation Review (canceled)**As noted above, the majority of the documentation was canceled. The API documentation review was conducted as part of the code review.
* **Verbal project summary** 1 Days 01.12.2012 01.12.2012  
  the final change to the project time line was a change to the project summary. Initially, the plan called for a written summary report of the project to be included in the TCIPG lab literature. As the full integration was postponed, so to was this written summary. It will be completed when the full integration work is completed later.  
    
  A verbal summary of work was given to lab management. This was done to inform lab management of the general availability of the integration framework, and a brief description of how to use it. This ensures that lab management is informed of the availability of the framework, and can be suggested to researchers who may not be familiar with the lab tools, but which may be able to help conduct research.

# Project Development

The development of this project was a resounding success. Despite changes to the schedule, which were mostly positive, and a tight deadline for the development of a prototype, all aspects of the project were completed and tested to the satisfaction of all stakeholders. In addition to completing the project, and making the tools available to researchers, the successful demonstration of the integration framework, as part of the KERK demonstrations, impressed and excited industry partners and TCIPG project sponsors.

The completion of this project is a key step in developing the tools necessary to provide a model research facility for the power gird, and will play a key role in future power grid research.

**Problems Encountered**

Throughout the development of this project, only two specific problems were encountered. The first, and most important, was the very tight deadline to completion of the prototype framework. The second, which had very little effect on the final success of the project, was an incorrect architectural design decision.

The tight deadline proved to be the biggest challenge in the completion of the project. As noted in the schedule above, only eight days were available from the start of the project to the hard deadline for the completion of the prototype. Under normal circumstances, such a tight deadline would be rejected, and stakeholders would be notified that an eight-day schedule for a new project is infeasible. However, as the lab management wished to display the fullest capabilities of the lab resources that were possible, we decided to commit to the deadline, and plan to work after hours as necessary.

During the week that development of the prototype was executed, several eighteen-hour days were used to reach completion of the prototype on time. In addition, one Saturday was spent fine-tuning the final prototype to meet the needs of the KERK project as requirement shifted during final testing of the KERK demonstration.

Both the last minute shifting of features and the long hours were expected from the commitment that was made. In the end, the prototype was completed on time, and successfully demonstrated the research lab capabilities. A video of the final presentation can be viewed on Youtube at <http://www.youtube.com/watch?v=BCNjIzvc_eg>.

During initial architectural design during the second refactoring stage of development, I planned to use the main-loop system provided by the tornado framework. By using tornado.ioloop, I would be freed from the requirement of maintaining input and output tracking. Tornado uses callback system that would enable handler methods to handle any input and output, as well as serve as the primary application main-loop. Unfortunately as I started development on the plugin system, it became clear that the level of complexity needed to make the tornado provided main-loop and the plugin system work together was beyond the desired complexity level. As such, development was halted while the architecture of the framework was readdress.

In the end, I implemented a custom main-loop that matched the needs of the plugin system. This decision worked out well, and, as it was written to the needs of the plugin system, meshed perfectly with the plugin architecture, as well as the input and output needs of the framework.

**Changes**

Aside from the architectural changes discussed above, there were several schedule changes, and well as some minor feature changes made during project development. The feature changes were made to better fit the needs of the KERK demonstration. The schedule changes came as a directive from management.

During the final testing of the framework, in preparation for the KERK presentation, it became clear that certain functionality of the KERK project required small changes to the framework prototype. There were two specific changes that were required.

During final testing, the KERK developers noted that the JSON formatted data stream being generated by RSCAD was incorrect. The KERK developers attempted to correct this formatting error with RSCAD, but ran into several roadblocks during their attempts to do so. They requested that the formatting corrections be made within the context of the framework, noting that doing so was not an optimal solution. After discussion, we all agreed that time did not permit finding the optimal solution, and an extra data processing step was added to the prototype. This data processing step read in the incorrectly formatted data, and reformatted it to meet the JSON format specification.

During later stages of development, I came concluded that retaining the data processing step as a utility function could prove to be beneficial to future project that use the framework. The final version of the framework include a module of utility function, the JSON reformat code was retained there.

The KERK project demonstration also required that the simulation that was running be stopped and swapped with another simulation during the course of the demonstration. The KERK developers looked at several options for doing this. The final decision was that the integration framework should respond to specific input, generate an RSCAD scriptlet that causes the swap, and wait for the simulation to restart. Such interaction with RSCAD was never planned, nor expected. However, the main-loop designed for the prototype was flexible enough that this new functionality was simple to add. Viewing the code, the make\_scriptlet() function implements this new feature.

As with the JSON formatting fixes, I found that this new functionality could be useful to future developers, and choose to retain it. In this instance, the functionality was fully integrated into the main-loop IO event system. By doing so, it gives experiment developers the option of using the interactive functionality, with very little performance impact if it isn’t needed.

Finally, there were several schedule changes. These changes came a result of changing priorities with the lab. While the framework is considered a key part of the overall lab strategy, the project documentation is not. Instead of a full written report of this project, management decided that a verbal overview of the project was sufficient. In addition, management concluded that a user manual was unnecessary. The rational for this decision is that the code was well written, and research developers should be able to use the framework with the built in help messages and the code itself as documentation. By canceling these portions of the project, the timeline to completion was shortened by about a month, and the project was completed early.

**Project Impact**

While there was many project specific goals, the most important objective of this project was to provide a tool to improve the TCIPG research lab. The lab’s goals of being the model facility for power grid and power grid security research drives and defines all of the projects conducted within the lab context. In this instance a framework was developed that integrates two the of the labs most important resources. By integrating these two resources, researchers now have the ability to create experiments that utilize the simulation capabilities of the Real Time Digital Simulator to simulate power grid infrastructure while using the DETER experimental framework for producing reproducible, well defined experiments. These two resources, together, enhance the research conducted in the TCIPG research lab.

**Conclusions**

The project was able to meet all of the requirements outlined by the KERK research developers. It also met the on going improvement goals of the TCIPG research lab. Bu successfully completing this project, power grid researchers has been given one more tool to use in conducting power grid research.

In addition, my work on this project and else were for the TCIPG research lab is personally rewarding. Though the average person will never know, the research that will use the results of this project will impact the everyday lives every person who receives electricity through the national power grid. To know that my work will have that level of impact provides the drive and passion I have for the work I do.

# Appendix 1: Usage Manual

usage: rscadstreamer.py [-h] [--delay SLEEPTIME] [--plugin-dir PATH]

[--plugin PLUGIN] [--script-file FILE] [--debug]

(--rscad rscad\_ip:port | --from-file FILE)

RSCAD Script interactions

optional arguments:

-h, --help show this help message and exit

--delay SLEEPTIME, -d SLEEPTIME

how long to sleep between cycles

--plugin-dir PATH, -p PATH

Location of plugins if not in PYTHON\_PATH

--plugin PLUGIN, -P PLUGIN

Plugin to load. Multiple --plugin/-P options allowed

--script-file FILE, -f FILE

--debug, -D Enable debugging output

Source:

Exacly one source must be selected

--rscad rscad\_ip:port, -R rscad\_ip:port

IP and Port of RSCAD

--from-file FILE, -F FILE

Read data stream from a file generated using --to-file

# Appendix 2: Acknowledgements

Special thanks goes out the following individuals who provided support, research assistance, and most importantly “rubber duck debugging”

Kate Morrow – Research Programmer: Debugging, Research Assistance, Development Support

Erich Heine – Research Programmer: Main Rubber Duck, Development Support, Design Support

David Rogers – Research Programmer: Research Assistance  
Tim Yardley – Director of Testbed Services: General Support; Freedom to develop the tools I feel are necessary for the research lab.